Python Başlangıç

3 Mart 2021 Çarşamba 19:51

İDE integrated development environment

Python'un Tarihsel Gelişimi
Bu programlama dili, 90'ların başında **Guido van Rossum** adlı Hollandalı bir programcı tarafından geliştirildi. Çoğu insan, bu programlama dilinin adını Python yılanından aldığını düşünür, adının Python olduğunu varsayar.
Ancak varsayımın aksine bu programlama dilinin adı python yılanından gelmez. Guido van Rossum, bu programlama dilini **The Monty Python** adlı bir İngiliz komedi grubundan **Monty Python'un Flying Circus** gösterisinden esinlenerek adlandırdı .

From <https://lms.clarusway.com/mod/lesson/view.php?id=2&pageid=2>

Bu dilin dünya çapında büyük bir geliştirici grubu var. Herhangi bir sorununuz varsa, her zaman diğer Python kullanıcılarından / geliştiricilerinden yardım isteyebilir veya **stackoverflow.com** gibi çeşitli sitelerde uygun bir yanıt bulabilirsiniz.

From <https://lms.clarusway.com/mod/lesson/view.php?id=2&pageid=2>

İpuçları:

- İfadeyi üç tırnakla çevrelemek: `"""..."""` veya `'''...'''` kodun, özellikle uzun metinlerde hata döndürmemesini sağlar.

From <https://lms.clarusway.com/mod/lesson/view.php?id=3&pageid=5>

Üzerinde Python kodları yazıp çalıştırabileceğimiz birkaç Entegre Geliştirme Ortamı (IDE) vardır. Sizin için birçok IDE seçeneği içeren **Anaconda Navigator (Anaconda3) paket** programını tercih ediyoruz.

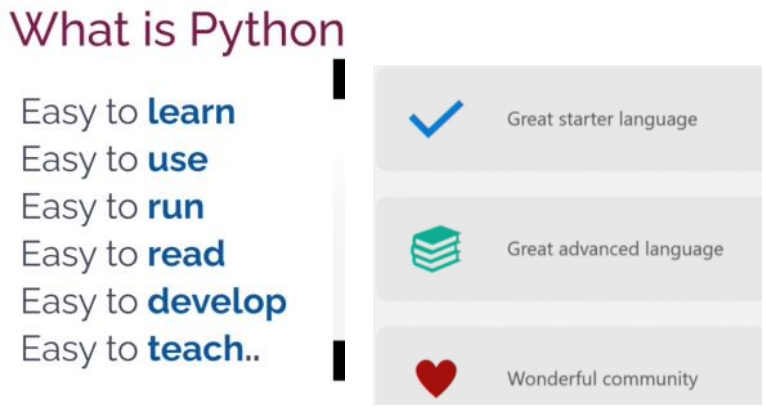
Anaconda'daki **IDE'ler arasında** , Python kodlarını üzerine yazmak için **Jupyter Lab'ı** kullanmayı tercih ediyoruz.

From <https://lms.clarusway.com/mod/lesson/view.php?id=2&pageid=3>

İLERİYE DÖNÜK TAVİSYE

terraform, ansible gibi UDEMY den ders al.

From <https://app.slack.com/client/T01P5K80UL9/D01PLSB1RBR



`__doc__` 2 alt çizgi ile dokuman açıklaması

What is Python?
Python is a programming language. It allows you to control the computer. The benefits of Pythons are that it is simple and easy, portable, extensible.

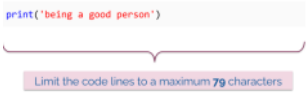
From <https://lms.clarusway.com/mod/hvp/view.php?id=508>

PEP 8

3 Mart 2021 Çarşamba 20:44

PEP, Python Geliştirme Önerisi anlamına gelir.
Python Enhancement Proposal

PEP , Python Geliştirme Önerisi anlamına gelir.
PEP 8, Python kodunuzu nasıl **daha okunaklı** yazacağınızla ilgili bir kodlama kuralı, bir dizi öneri **.**
Başka bir deyişle, PEP 8, ana Python dağıtımındaki standart kitaplığı oluşturan Python kodu için kodlama kuralları veren bir belgedir.



Tüm satırları maksimum boşluklar dahil **79 karakterle** sınırlayın .
Daha az yapısal kısıtlamayla (belge dizileri veya yorumlar) akan uzun metin blokları için satır uzunluğu **72 karakterle** sınırlandırılmalı

- Aşağıdaki durumlarda gereksiz **boşluklardan** kaçının :
  Hemen parantez, parantez veya kaşlı ayraç içinde:
  EVET : spam(meat[1], {milk: 2}), **HAYIR** : spam(meat[1], { milk: 2 })

  Sondaki virgül ile aşağıdaki yakın parantez arasında:
  EVET : df[0,]veya foo = (2,), **HAYIR** : df[0,]veyafoo = (2,)

  Virgül, noktalı virgül veya iki nokta üst üste işaretinden hemen önce:
  EVET : if y == 3: print x, y; x, y = y, x, **HAYIR** : if y == 3 : print x , y ; x , y = y , x

  Bir işlev çağrısının bağımsız değişken listesini başlatan açık parantezden hemen önce:
  EVET : print('peace'), **HAYIR** : print ('peace')

  Bir atama (veya başka) operatörünü bir başkasıyla hizalamak için birden fazla boşluk:

EVET	HAYIR
x = 3	x = 3
y = 4	y =mmmmm4
long_vars = 5	long_vars = 5

- Herhangi bir yerde arka arkaya boşluk bırakmayın. Genellikle görünmez olduğu için kafa karıştırıcı olabilir: ör. Bir boşluk ve ardından gelen bir ters eğik çizgi ve bir satırsonu bir satır devam işaretçisi olarak sayılmaz.
- Bu ikili operatörleri her iki tarafta da tek bir boşlukla çevreleyin: atama (=), artırılmış atama (+=, -=, etc.), karşılaştırmalar (==, <, >, !=, <>, <=, >=, in, not in, is, is not), Boole'lar (and, or, not).

From <https://lms.clarusway.com/mod/lesson/view.php?id=4&pageid=7>

Jupyter lab
Açık Kaynak kodlu ve Hücre Yapılıdır. Colab de jupiiter tabanlıdır.

Tek tırnaklar kafasını karıştırdı

```
print('I'am happy')
```

```
  File "<ipython-input-6-2854e1a812e0>", line 1
    print('I'am happy')
           ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

Tırnaklar farklı olduğu için hata yok

```
print('Arkadaş bana "merhaba" dedi')
```

```
Arkadaş bana "merhaba" dedi
```

```
print('''Lorem ipsum dolor sit amet,
 fusce ut quisque neque donec,
massa metus amet, luctus inceptos.''')

print("""Praesent a, mi mattis velit metus,
accumsan adipiscing ipsum sit justo
penatibus, amet mauris non tempus justo.""")
```

Triple use of 'single' and "double" quotes

Yada hiç bunlarla uğraşma direk 3 tırnak kullan.
''' ''' veya """ """

ÇIKTISI : 'we..... family"

```
print('''We should have enough time for our family"''')
```

```
print('3.14')
print(3.14)
```

String [Text]

Ondalık sayı

```
print('.'')
```

```
  File "<ipython-input-21-f523f153590c>",
    print('.'')
           ^
SyntaxError: invalid syntax
```

```
print('.''')
```

```
.
```

```
print('''asd'''''22''')
```

```
asd22
```

```
print('.''2')
```

```
.2
```

```
print("asd"''''22''')
```

```
asd22
```

İlk 3lü tek tırnaktan sonra bir sonraki 3lü tek tırnağı aradı.
Sonrası için tekrardan hesap ediyor.
En sondaki bu pembe tırnak tek kaldı

```
print(''''1''''')
```

```
  File "<ipython-input-1-c
    print(''''1''''')
                       ^
SyntaxError: EOL while sca
```

```
print('''1''''')
```

1 tırnak daha ekledim.

```
'1
```

Yine İlk 3lü tek tırnaktan sonra bir sonraki 3lü tek tırnağı aradı.
2 tane tekli ile bir grup oluşmuş oldu.

```
print('''1''''''')
```

1 tırnak daha ekledim.
Yine İlk 3lü tek tırnaktan sonra bir sonraki 3lü tek tırnağı aradı.
Sondaki 3lü ile işlem yapılamadığından içi boş tek tırnaklar hata verdi.

```
  File "<ipython-input-3-8b2afa12a27b>", line 1
    print(''''1''''''')
                       ^
```

Python Mülakat Soruları

From <https://lms.clarusway.com/mod/hvp/view.php?id=517&forceview=1>

Python'da boole nedir?

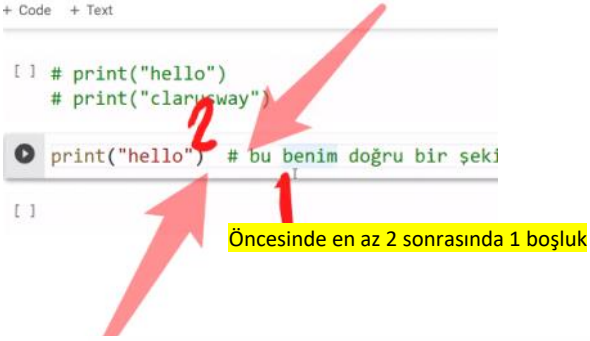
```
banana_price = 2
total_amount_bought = 5
print(banana_price * total_amount_bought)
```

Check

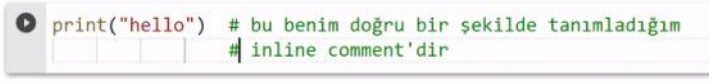
	Expected	Got	
✔	10	10	✔

x>y çalışır ama PEP8e göre yanlış

ctrl+shift+7 ile tüm seçilenler yorum oluyor.



Öncesinde en az 2 sonrasında 1 boşluk



Hizalama yapabilirsin.

Comments and Docstrings

HASH karakter

Başlamak için, □□ #'dan sonra bir boşluk olmalı ve satır içi yorumlarda kodun sonu ile □□ # arasında en az iki boşluk olmalıdır.

From <https://lms.clarusway.com/mod/lesson/view.php?id=5&pageid=10>

Doctrings- Dokümanlar
Normal yorumlardan farklı olarak - dokümante ettikleri işlevin veya modülün bir niteliği olarak saklanır, bu da onlara programlı olarak erişebileceğiniz anlamına gelir. Docstring açıklayıcı bir kod metni olarak çalışır ve üçlü tırnak arasında yazılmalıdır.

From <https://lms.clarusway.com/mod/hvp/view.php?id=512&forceview=1>

Bir docstring nasıl görünmeli?

- **docstring** büyük harfle başlamalıdır.
- İlk satır kısa bir açıklama olmalıdır.
- **docstring'de** daha fazla satır varsa, ikinci satır boş olmalı, özeti görselin geri kalanından görsel olarak ayırmalısınız.
- Aşağıdaki satırlar, nesnenin arama kurallarını, yan etkilerini vb. açıklayan bir veya daha fazla paragraf olmalıdır.

Docstring Tanımlama:, yöntemin veya işlev bildiriminin hemen altında ''' yada """ ifadesiyle bildirilir. Tüm işlevlerin bir belge olması gerekir.

Docstring Ulaşım: Docstrings'e nesnenin **__doc__** yöntemi kullanılarak veya yardım işlevini kullanarak erişilebilir.
Aşağıdaki örnek, bir dokümanın nasıl bildirileceğini ve erişileceğini göstermektedir.

```
def ornekfunc():
    """Örnek bir fonksiyon olarak yapıldı."""
    return None

print ("Kullanımı __doc__:")
print (ornekfunc.__doc__)

print ("help kullanımı:")
help(ornekfunc)
```

Displaying the docstring of a function :

```
print(function_name.__doc__)
```

Normally, when we want to call docstring of a function or module to read, we use .__doc__ (the keyword doc enclosed by double underscores) syntax.

```
Hi, I am the docstring of this code.
If you need any information about this function or
module, read me.
It can help you understand how the module or function
works.
```

Alt çizgiler 2 şer tane

input :

```
print(print.__doc__)
```

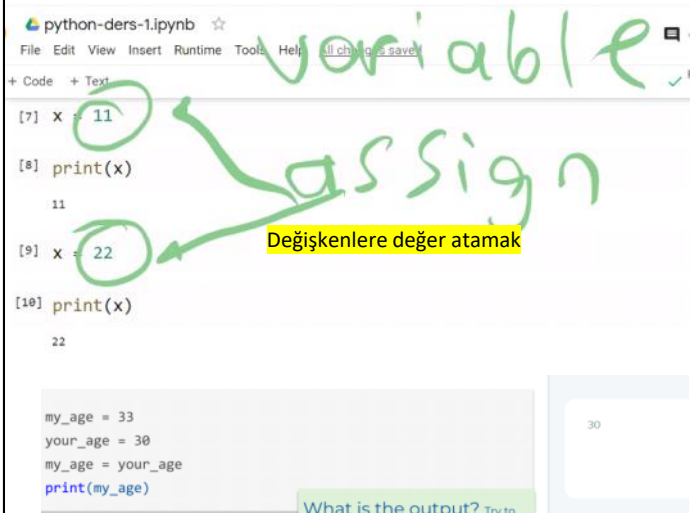
output :

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
```

```
variable name = value
planet = 'jupyter'
price = 140
pi_number = 3.14
```

Yazılar tırnak içinde olmalı.
Eşittir in yanlarında 1 boşluk olmalı



Değişkenlere değer atamak

```
[22] print(x)
     1

[23] y = 2
     z = 3

[24] print(x,y,z)
     1 2 3

[25] print(x+y)
```

x,y,z arasında boşluk olmayacak

```
your_age = 30
my_age = your_age
print(my_age)
```

What is the output? Try to figure out in your mind...

+ Add another response

İşlem sırası ile yaş değeri değişmiş oldu

```
[25] print(x+y)

     3
```

```
x = 10
print(x)
x = 11
print(x)
x = 12
print(x)
x = 13
print(x)
```

```
10
11
12
13
```

```
print(x)
```

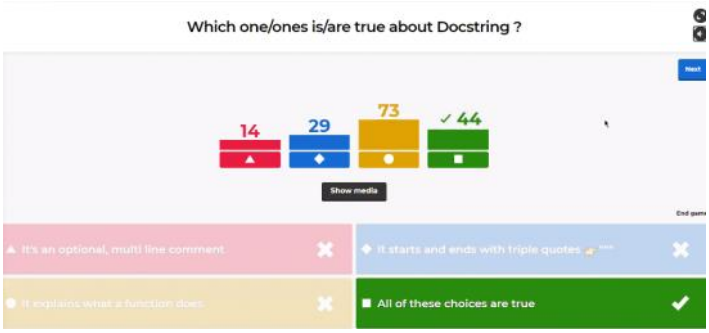
```
13
```

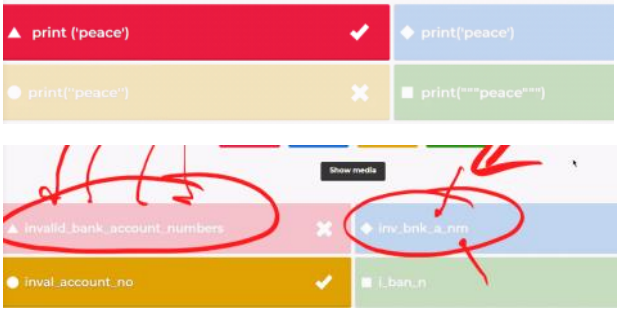
```
[14] xx = 11
     yy = 21
     zz = 15
```

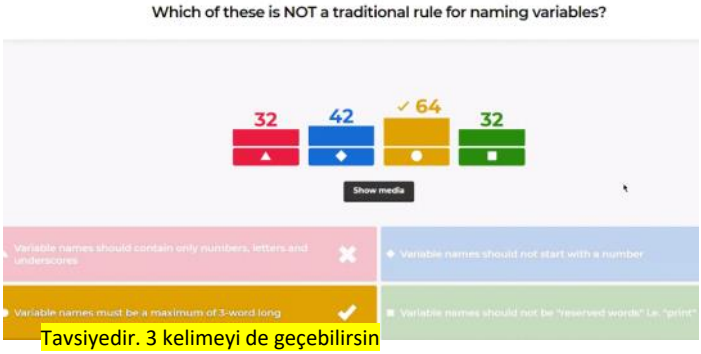
```
[15] xx = yy
     zz = xx
```

```
print(xx)
print(yy)
print(zz)
```

```
21
21
21
```







Tavsiyedir. 3 kelimeyi de geçebilirsin

Some basic naming conventions

- Do not use `'1'` **(lowercase letter "L")** character variable.

```
l = 3.14  # This is lowercase letter el
I = 3.14  # This is uppercase letter eye
1 = 'something wrong'  # This is number one.
```

- Tek başına küçük L harfi kullanılmıyor.
- Büyük I (ı) harfi de kullanılmıyor.
- 1 ile (sayı ile) başlanmaz.

- Do not use `'O'` **(uppercase letter "O")** as single character variable.

```
time_O = '3.14'  # This is uppercase letter "O"
time_0 = '3.14'  # This is number zero
```

- O harfini de kullanmıyoruz. Programda çalışır ama karışabilir.

```
[22] O = 0

     time_0 = 11
     time_O = 12
```

- Do not use specific Python keywords such as :

```
False   class     finally   is        return
None    continue  for       lambda    try
True    def       from      nonlocal  while
and     del       global    not       with
```

Bu kelimeleri de değişken atama da kullanma. Zaten renklerle uyarı veriyor

```
[24] import = 5
```

```
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
```

```
[24] import = 5

  File "<ipython-input-24-3c1c1f1c3721>", line 1
    import = 5
           ^
SyntaxError: invalid syntax

SEARCH STACK OVERFLOW
```

```
[25] x = 0
     X = 1
     print(x)

0
```

CASE SENSİTİVE - harfe duyarlı
alttaki x büyük harf olmuş. Onda hata verdi.
çıktı sadece küçük harfi gösterdi.

```
if
in
with
False
```

Küçük harf ile false yazılsa idi renkledirme olmazdı.
Çünkü CASE SENSİTİVE. Büyük- küçük harf duyarlı

Pythonic Rules

Examine these samples carefully

2me	😔	data4me	😀
Big boss	😔	first!	😔
big-boss	😔	xy#@!v	😔
last_name	😀	$price	😔

```
4me = 2

  File "<ipythor
    4me = 2
        ^
SyntaxError: inv

SEARCH STACK OVE
```

```
[37] 2me = 5

  File "<ipython-i
    2me = 5
        ^
SyntaxError: inval

SEARCH STACK OVERFL
```

```
[23] me!you = 5

  File "<ipythor
    me!you = 5
           ^
SyntaxError: inv

SEARCH STACK OVE
```

```
[38] Ali bey = baba

  File "<ipython-i
    Ali bey = baba
            ^
SyntaxError: invali

SEARCH STACK OVERFLO
```

```
[24] small home = not
  File "<ipython-input-24-fca4
    small home = not
               ^
```

```
big-band = 6
```

```
[40] sen2 = 0

[35] champGS = 0

[41] _kim = 4

[42] -kimse = 5

  File "<ipytho
    -kimse = 5
           ^
SyntaxError: ca
```

the list of mathematics exam scores

- Good samples :
  - math_scores
  - score_maths

Keywords

3 Mart 2021 Çarşamba 19:47

Python Cheat Sheet: Keywords

"A puzzle a day to learn, code, and play" → Visit finxter.com

Keyword	Description	Code example
`False, True`	Data values from the data type Boolean	`False == (1 > 2), True == (2 > 1)`

`and, or, not`	Logical operators:
	`(x and y)` → both x and y must be True
	`(x or y)` → either x or y must be True
	`(not x)` → x must be false
		`x, y = True, False`
		`(x or y) == True  # True`
		`(x and y) == False  # True`
		`(not y) == True  # True`

`break`	Ends loop prematurely	`while(True):`
		`  break # no infinite loop`
		`print("hello world")`

`continue`	Finishes current loop iteration	`while(True):`
		`  continue`
		`  print("43") # dead code`

`class`
 	
`def`
	Defines a new class → a real-world concept (object oriented programming)
	
	Defines a new function or class method. For latter, first parameter ("self") points to the class object. When calling class method, first parameter is implicit.
		`class Beer:`
		`  def __init__(self):`
		`  self.content = 1.0`
		`  def drink(self):`
		`  self.content = 0.0`
		
		`becks = Beer() # constructor - create class`
		`becks.drink() # beer empty: b.content == 0`

`if, elif, else`	Conditional program execution: program starts with "if" branch, tries the "elif" branches, and finishes with "else" branch (until one branch evaluates to True).	`x = int(input("your value: "))`
		`if x > 3: print("Big")`
		`elif x == 3: print("Medium")`
		`else: print("Small")`

`for, while`	`# For loop declaration`
	`for i in [0,1,2]:`
	`  print(i)`
		`# While loop - same semantics`
		`j = 0`
		`while j < 3:`
		`  print(j)`
		`  j = j + 1`

`in`	Checks whether element is in sequence	`42 in [2, 39, 42] # True`

`is`	Checks whether both elements point to the same object	`y = x = 3`
		`x is y # True`
		`[3] is [3] # False`

`None`	Empty value constant	`def f():`
		`  x = 2`
		`f() is None # True`

`lambda`	Function with no name (anonymous function)	`(lambda x: x + 3)(3) # returns 6`

`return`	Terminates execution of the function and passes the flow of execution to the caller. An optional value after the return keyword specifies the function result.	`def incrementor(x):`
		`  return x + 1`
		`incrementor(4) # returns 5`

finxter

Data-Structures

3 Mart 2021 Çarşamba 19:47

Python Cheat Sheet: Basic Data Types

"A puzzle a day to learn, code, and play" → Visit finxter.com

	Description	Example

Boolean	The Boolean data type is a truth value, either `True` or `False`.
	
	The Boolean operators ordered by priority:
	`not x` → "if x is False, then x, else y"
	`x and y` → "if x is False, then x, else y"
	`x or y` → "if x is False, then y, else x"
	
	These comparison operators evaluate to `True`:
	`1 < 2 and 0 <= 1 and 3 > 2 and 2 >=2 and 1 == 1 and 1 != 0 # True`
		`## 1. Boolean Operations`
		`x, y = True, False`
		`print(x and not y) # True`
		`print(not x and y or x) # True`
		
		`## 2. If condition evaluates to False`
		`if None or 0 or 0.0 or '' or [] or {} or set():`
		`  # None, 0, 0.0, empty strings, or empty`
		`  # container types are evaluated to False`
		`  print("Dead code") # Not reached`

Integer, Float	An integer is a positive or negative number without floating point (e.g. `3`). A float is a positive or negative number with floating point precision (e.g. `3.14159265359`).
	
	The '`//`' operator performs integer division. The result is an integer value that is rounded toward the smaller integer number (e.g. `3 // 2 == 1`).
		`## 3. Arithmetic Operations`
		`x, y = 3, 2`
		`print(x + y) # = 5`
		`print(x - y) # = 1`
		`print(x * y) # = 6`
		`print(x / y) # = 1.5`
		`print(x // y) # = 1`
		`print(x % y) # = 1s`
		`print(-x) # = -3`
		`print(abs(-x)) # = 3`
		`print(int(3.9)) # = 3`
		`print(float(3)) # = 3.0`
		`print(x ** y) # = 9`

String	Python Strings are sequences of characters.
	
	The four main ways to create strings are the following.
	
	1. Single quotes
	`'Yes'`
	2. Double quotes
	`"Yes"`
	3. Triple quotes (multi-line)
	`"""Yes`
	`We Can"""`
	4. String method
	`str(5) == '5' # True`
	5. Concatenation
	`"Ma" + "hatma" # 'Mahatma'`
	
	These are whitespace characters in strings.
	• Newline `\n`
	• Space `\s`
	• Tab `\t`
		`## 4. Indexing and Slicing`
		`s = "The youngest pope was 11 years old"`
		`print(s[0])  # 'T'`
		`print(s[1:3])  # 'he'`
		`print(s[-3:-1])  # 'ol'`
		`print(s[-3:])  # 'old'`
		`x = s.split()  # creates string array of words`
		`print(x[-3] + " " + x[-1] + " " + x[2] + "s")`
		`  # '11 old popes'`
		
		`## 5. Most Important String Methods`
		`y = "  This is lazy\t\n  "`
		`print(y.strip()) # Remove Whitespace: 'This is lazy'`
		`print("DrDre".lower()) # Lowercase: 'drdre'`
		`print("attention".upper()) # Uppercase: 'ATTENTION'`
		`print("smartphone".startswith("smart")) # True`
		`print("smartphone".endswith("phone")) # True`
		`print("another".find("other")) # Match index: 2`
		`print("cheat".replace("ch", "m")) # 'meat'`
		`print(','.join(["F", "B", "I"])) # 'F,B,I'`
		`print(len("Rumpelstiltskin")) # String length: 15`
		`print("ear" in "earth") # Contains: True`

finxter

Complex-Data-Types

3 Mart 2021 Çarşamba 19:47

Python Cheat Sheet: Complex Data Types

"*A puzzle a day to learn, code, and play*" → Visit finxter.com

	Description	Example
List	A container data type that stores a sequence of elements. Unlike strings, lists are mutable: modification possible.	`l = [1, 2, 2]` `print(len(l)) # 3`

Adding elements	Add elements to a list with (i) append, (ii) insert, or (iii) list concatenation.
	The append operation is very fast.
		`[1, 2, 2].append(4) # [1, 2, 2, 4]`
		`[1, 2, 4].insert(2,2) # [1, 2, 2, 4]`
		`[1, 2, 2] + [4] # [1, 2, 2, 4]`

Removal	Removing an element can be slower.	`[1, 2, 2, 4].remove(1) # [2, 2, 4]`
Reversing	This reverses the order of list elements.	`[1, 2, 3].reverse() # [3, 2, 1]`
Sorting	Sorts a list. The computational complexity of sorting is linear in the no. list elements.	`[2, 4, 2].sort() # [2, 2, 4]`
Indexing	Finds the first occurence of an element in the list & returns its index. Can be slow as the whole list is traversed.	`[2, 2, 4].index(2) # index of element 4 is "0"` `[2, 2, 4].index(2,1) # index of element 2 after pos 1 is "1"`

Stack	Python lists can be used intuitively as stacks via the two list operations append() and pop().	`stack = [3]`
		`stack.append(42) # [3, 42]`
		`stack.pop() # 42 (stack: [3])`
		`stack.pop() # 3 (stack: [])`

Set	A set is an unordered collection of unique elements ("at-most-once").	`basket = {'apple', 'eggs', 'banana', 'orange'}` `same = set(['apple', 'eggs', 'banana', 'orange'])`
Dictionary	The dictionary is a useful data structure for storing (key, value) pairs.	`calories = {'apple' : 52, 'banana' : 89, 'choco' : 546}`

Reading and writing elements	Read and write elements by specifying the key within the brackets. Use the keys() and values() functions to access all keys and values of the dictionary.	`print(calories['apple'] < calories['choco']) # True`
		`calories['cappu'] = 74`
		`print(calories['banana'] < calories['cappu']) # False`
		`print('apple' in calories.keys()) # True`
		`print(52 in calories.values()) # True`

Dictionary Looping	You can access the (key, value) pairs of a dictionary with the `items()` method.	`for k, v in calories.items():` `  print(k) if v > 500 else None # 'chocolate'`

Membership operator	Check with the 'in' keyword whether the set, list, or dictionary contains an element. Set containment is faster than list containment.	`basket = {'apple', 'eggs', 'banana', 'orange'}`
		`print('eggs' in basket) # True`
		`print('mushroom' in basket) # False`

List and Set Comprehension	List comprehension is the concise Python way to create lists. Use brackets plus an expression, followed by a for clause. Close with zero or more for or if clauses.
	
	Set comprehension is similar to list comprehension.
		`# List comprehension`
		`l = [('Hi ' + x) for x in ['Alice', 'Bob', 'Pete']]`
		`print(l) # ['Hi Alice', 'Hi Bob', 'Hi Pete']`
		`l2 = [x * y for x in range(3) for y in range(3) if x>y]`
		`print(l2) # [0, 0, 2]`
		`# Set comprehension`
		`squares = { x**2 for x in [0,2,4] if x < 4 } # {0, 4}`

finxter

- Ek Özet

4 Mart 2021 Perşembe 22:23

General Information about Python

What is Python?

Tahmin edebileceğiniz gibi, Python bir programlama dilidir (Java, C ++, R, Ruby vb.). Diğer programlama dilleri gibi, makineyi önünüzde, yani bilgisayarı kontrol etmenizi sağlar.

IT endüstrisi, Veri bilimi uygulamaları ile patlama yaşıyor:

-Yapay zeka
-Derin öğrenme
-Makine öğrenimi algoritmaları.

Python, bu alanda en yaygın kullanılan teknolojidir. Yeni çağ uygulamalarıyla, bir Python geliştiricisine olan talep de arttı.

Google arama trendlerine göre 2019 yılında Python, tüm programlama dilleri arasında en popüler aranan terimdir. Tiobe.com'un programlama dilleri dizinine göre, Python en hızlı büyüyen dildir.

GitHub'ın popüler kod paylaşım sitesinin kullanımıyla ilgili 2019 State of the Octoverse raporuna göre, Python GitHub'ın ikinci en popüler dili haline geldi ve Java'yı ilk kez geride bıraktı.

Bu dil ile programlar çok hızlı geliştirilebilir. Ek olarak, Python programlama dilinin basit ve temiz sözdizimi, onu birçok programcı tarafından tercih edilen bir dil haline getirmiştir.Başkaları tarafından yazılan programları yazmak ve okumak kolaydır. Bu nedenle, özellikle Veri Biliminde yaygın olarak kullanılmaktadır ve son yıllarda çok sayıda talep almıştır.

1.ÖDEV Python kullanan 1.nci sınıf şirketler;

Industrial Light and Magic(Endüstriyel Işık ve Büyü), google,facebook, instagram, spotify, quora, netflix, dropbox, reddit

Diğerlerinden ayıran özellikleri: Basit ,kolay, taşınabilir, genişletilebilir , öğretilebilir

General Information about Python

*The Programming Language of the Agile Era (Optional)

Bu bölümdeki içerikler resmi broşür kullanılarak geliştirilmiştir (Python Broşür Cilt)

2. Baskı,Python Yazılım Vakfı (PSF) tarafından Python'u tanıtmak için hazırlanan Case Studies & Success Stories). Broşürün tamamına buradan ulaşabilirsiniz.

*Programming with Python:

Yazılım kalitesi, endüstri ve bilimde başarı için hayati bir bileşendir. Her yerde bulunan IT sistemleri, küresel ekonominin iş süreçlerini kontrol eder.Giderek daha güçlü bilgisayarlar ve gelişmiş algoritmalar, yeni bilimsel keşifler için bir platform sağlar.Ve küresel iletişim, akıllı yazılım olmadan düşünülemez. Müşteri

yarışında, rakiplerinden daha hızlı pazara girenlerin ilk sırayı alması.Daha iyi ve daha yaratıcı çözümler, yeni zorluklara anında yanıt verme becerisiyle birlikte yarışı yönlendirir. Normalde gerekli olan sürenin bir kısmında güvenli ve güvenilir programlar yazmak sizi bitiş çizgisine ilk adım atmanızı sağlar.

*The Programming Language of the Agile Era :

Çeviklik, zamanımızın ayırt edici özelliğidir ve Python, çevik çağın programlama dilidir. Python evrensel programlama dili, IT departmanının turboşarjıdır.Java veya C gibi diğer modern programlama dilleriyle karşılaştırıldığında Python, bir dizi farklı nedenden ötürü önemli ölçüde daha kısa sürelerde üstün sonuçlar elde eder.

Örneğin, Python çok yalın bir programlama dilidir. Python programları, diğer modern programlama dillerinde yazılmış kodlardan çok daha kısadır.Sonuç olarak, hem geliştirme süreleri hem de bakım maliyetleri büyük ölçüde azalır. Daha az kod,

daha az hata anlamına gelir, yani bu hataları tanımlama ve ortadan kaldırma maliyeti de azalır.

Python Paket Dizinindeki kapsamlı bir standart kitaplık ve binlerce ek kitaplık, geliştiricilere neredeyse her gereksinimi karşılamak için uygulamalarına kolayca entegre edebilecekleri yüksek kaliteli çözümler sunar.

Bu şekilde Python, başka yerlerde daha verimli kullanım için tahsis edilebilecek geniş kaynakları serbest bırakır.

*The Master Key for System Integration :

Python, sistem entegrasyonu için benzersiz avantajlar sunar. Bir yandan, neredeyse tüm üçüncü taraf sistemlerin entegre edilebileceği çok sayıda Python kitaplığı vardır.Öte yandan, diğer birçok programlama dilinin kitaplıkları da Python'da kullanılabilir.

Python uygulamaları programlandıktan sonra, bir Python yorumlayıcısının bulunduğu tüm işletim sistemlerinde çalışabilir ve işletim sistemine özel uygulamaların maliyetini önemli ölçüde azaltır.

*The Language that has Changed Everything :

Python, 20 yılı aşkın süredir endüstride, hizmet sektöründe ve ayrıca araştırma ve bilimde çok çeşitli farklı gereksinimleri karşılamak için bir programlama dili olarak dünya çapında başarıyla kullanılmaktadır.Bu zamanda dil birçok şeyi değiştirdi.

Python programlama dilinin öğrenilmesi kolaydır. Kullanıcılar ve geliştiriciler arasındaki sınırları bulanıklaştırdı.Artan sayıda bilim adamı, mühendis, finans uzmanı ve çok az programlama deneyimi olan diğerleri, belirli karmaşık teknik sorunları çözmek için Python'u kullanıyor.

*Historical Development of Python:

Bu programlama dili, 90'ların başında Guido van Rossum adlı Hollandalı bir programcı tarafından geliştirildi. Çoğu insan, bu programlama dilinin ismini Python yılanından aldığını düşünür ve isminin Python olduğunu varsayar.

Ancak varsayımın aksine bu programlama dilinin adı python yılanından gelmemektedir.Guido van Rossum, bu programlama dilini The Monty Python adlı bir İngiliz komedi grubunun Monty Python'un Flying Circus gösterisinden esinlenerek adlandırdı.

Durum böyle olsa da, pek çok yerde yılan tipli Python programlama dilinin logosu adeta bir gelenek haline geldi.

Bu dilin dünya çapında büyük bir geliştirici grubu var. Herhangi bir sorununuz varsa, her zaman diğer Python kullanıcılarından / geliştiricilerinden yardım isteyebilir veya stackoverflow.com gibi çeşitli sitelerde uygun bir yanıt bulabilirsiniz.

*Review of Tools & Installations:

Python 3.X sürümü 2008'de piyasaya sürüldü. Python'u önceki sürümlere göre daha okunabilir ve tutarlı hale getiriyor. Bu kurs boyunca, Python 3.X'in (şu anda Python 3.9.0) en son sürümlerini kullanacağız.

Üzerinde Python kodları yazıp çalıştırabileceğimiz birkaç Entegre Geliştirme Ortamı (IDE) vardır. Sizin için çeşitli IDE seçenekleri içeren Anaconda Navigator (Anaconda3) paket programını tercih ediyoruz.Anaconda'yı isteğe bağlı olarak bilgisayarınıza kurabilirsiniz. Şu anda Python 3.7, Anaconda paketinde mevcuttur.

Anaconda'daki IDE'ler arasında, üzerine Python kodlarını yazmak için Jupyter Lab'ı kullanmayı tercih ediyoruz.Jupyter Lab, canlı kod, denklemler, görselleştirmeler ve anlatı metni içeren belgeler oluşturmanıza ve paylaşmanıza olanak tanıyan açık kaynaklı bir web uygulamasıdır. Jupyter Lab'e çok benzeyen Jupyter Notebook da kullanılabilir.

Anaconda dışında elbette Python Shell (IDLE) Python için temel bir yorumlayıcı araç olarak da kullanılabilir.

*Throughout this course :

- Her ders sayfasında kodunuzu yazıp çalıştırmanız için size Playground modülü sağlayacağız. Bu nedenle daha önce de belirttiğimiz gibi Anaconda Paketini kurmanıza gerek yoktur, isteğe bağlıdır.
- Ek olarak, sınıf içi oturumlar sırasında Google Colab uygulaması dahil tüm Jupyter Not Defteri tabanlı IDE'leri kullanacağız.
- Python kodlarına göre birkaç Scratch alıştırmasına sahip olacaksınız. Scratch alıştırmalarının nasıl çözüleceğine ilişkin açıklayıcı bir videoyu aşağıda bulabilirsiniz.