



Introduction to Programming

Lesson 7

Outline

- random module
- list comprehensions
- String-Encoding/Byte-Decoding

Lecture Code

goo.gl/PVe0KE

random / պատահական

Պատահական թվերը պետք են գալիս

- scientific computing
- financial simulations
- cryptography
- computer games

random / պատահական

Պատահական թվերը պետք են գալիս

- scientific computing
- financial simulations
- cryptography
- computer games

Իսկական պատահական թվեր դժվար է ստանալ/գենեռացնել

random / պատահական

Պատահական թվերը պետք են գալիս

- scientific computing
- financial simulations
- cryptography
- computer games

Իսկական պատահական թվեր դժվար է ստանալ/գենեռացնել

Օգտագործում են **pseudorandom number generator**, որում

- numbers only appear to be random
- they are really generated using a deterministic process

random / պատահական

Պատահական թվերը պետք են գալիս

- scientific computing
- financial simulations
- cryptography
- computer games

Իսկական պատահական թվեր դժվար է ստանալ/գենեռացնել

Օգտագործում են **pseudorandom number generator**, որում


- numbers only appear to be random
- they are really generated using a deterministic process

library module random ունի pseudo random number generator և տարբեր օգտակար functions: dir(random)

random module

Function `randrange()` returns
a “random” integer *տրված*
տիրույթից

1-ից մինչև 7, 7-ը չներառած




```
>>> import random
>>> random.randrange(1, 7)
2
>>>
```


random module

1-ից մինչև 7, 7-ը չներառած

Function `randrange()` returns
a “random” integer տրված
տիրույթից

Example: simulate the throws
of a die (սիմուլյացիա անել
զառի նետումը)



```
>>> import random
>>> random.randrange(1, 7)
2
>>>
```

random module

Function **randrange()** returns
a “random” integer տրված
տիրույթից

Example: simulate the throws
of a die (սիմուլյացիա անել
զառի նետումը)

```
>>> import random
>>> random.randrange(1, 7)
2
>>> random.randrange(1, 7)
1
>>>
```

random module

Function **randrange()** returns
a “random” integer տրված
տիրույթից

Example: simulate the throws
of a die (սիմուլյացիա անել
զառի նետումը)

```
>>> import random
>>> random.randrange(1, 7)
2
>>> random.randrange(1, 7)
1
>>> random.randrange(1, 7)
4
>>>
```

random module

Function `randrange()` returns
a “random” integer տրված
տիրույթից

Example: simulate the throws
of a die (սիմուլյացիա անել
զառի նետումը)

```
>>> import random
>>> random.randrange(1, 7)
2
>>> random.randrange(1, 7)
1
>>> random.randrange(1, 7)
4
>>> random.randrange(1, 7)
2
>>>
```

random module

Function `randrange()` returns a “random” integer տրված տիրույթից

Example: simulate the throws of a die (սիմուլյացիա անել զառի նետումը)

Function `uniform()` returns a “random” float number տրված տիրույթից

```
>>> import random
>>> random.randrange(1, 7)
2
>>> random.randrange(1, 7)
1
>>> random.randrange(1, 7)
4
>>> random.randrange(1, 7)
2
>>> random.uniform(0, 1)
0.19831634437485302
>>> random.uniform(0, 1)
0.027077323233875905
>>> random.uniform(0, 1)
0.8208477833085261
>>>
```

random module

```
>>> names = ['Ann', 'Bob', 'Cal', 'Dee', 'Eve', 'Flo',  
             'Hal', 'Ike']
```

random module

```
>>> names = ['Ann', 'Bob', 'Cal', 'Dee', 'Eve', 'Flo',  
'Hal', 'Ike']  
>>> import random  
>>> random.shuffle(names)  
>>> names  
['Hal', 'Dee', 'Bob', 'Ike', 'Cal', 'Eve', 'Flo', 'Ann']
```

random module

```
>>> names = ['Ann', 'Bob', 'Cal', 'Dee', 'Eve', 'Flo',  
'Hal', 'Ike']  
>>> import random  
>>> random.shuffle(names)  
>>> names  
['Hal', 'Dee', 'Bob', 'Ike', 'Cal', 'Eve', 'Flo', 'Ann']  
>>> random.choice(names)  
'Bob'  
>>> random.choice(names)  
'Ann'  
>>> random.choice(names)  
'Cal'  
>>> random.choice(names)  
'Cal'
```


random module

```
>>> names = ['Ann', 'Bob', 'Cal', 'Dee', 'Eve', 'Flo',  
'Hal', 'Ike']  
>>> import random  
>>> random.shuffle(names)  
>>> names  
['Hal', 'Dee', 'Bob', 'Ike', 'Cal', 'Eve', 'Flo', 'Ann']  
>>> random.choice(names)  
'Bob'  
>>> random.choice(names)  
'Ann'  
>>> random.choice(names)  
'Cal'  
>>> random.choice(names)  
'Cal'  
>>> random.sample(names, 3)  
['Ike', 'Hal', 'Bob']  
>>> random.sample(names, 3)  
['Flo', 'Bob', 'Ike']  
>>> random.sample(names, 3)  
['Ike', 'Ann', 'Hal']
```

list comprehension/ըմբռնում

```
[expression for element in list]
```

```
>>>
```

list comprehension/ըմբռնում

```
[expression for element in list]
```

≈ համարժեք է

```
for element in list:  
    expression
```

```
>>>
```

list comprehension/ըմբռնում

```
[expression for element in list]
```

≈ համարժեք է

```
for element in list:  
    expression
```

expression-ը կիրառվում է list-ի ամեն անդամի վրա

```
>>>
```

list comprehension/ըմբռնում

```
[expression for element in list]
```

≈ համարժեք է

```
for element in list:  
    expression
```

expression-ը կիրառվում է list-ի ամեն անդամի վրա

```
>>>  
>>> squares = []  
>>> for x in range(10):  
...     squares.append(x**2)  
...  
>>> squares  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

list comprehension/ըմբռնում

```
[expression for element in list]
```

≈ համարժեք է

```
for element in list:  
    expression
```

expression-ը կիրառվում է list-ի ամեն անդամի վրա

```
>>>  
>>> squares = []  
>>> for x in range(10):  
...     squares.append(x**2)  
...  
>>> squares  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  
>>> squares = [x**2 for x in range(10)]  
>>> squares  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]7
```

filtered list comprehension

```
[ expression for name in list if filter ]
```

filtered list comprehension

```
[ expression for name in list if filter ]
```

≈ `hwfwpdtp t`

```
for element in list:  
    if filter:  
        expression
```


filtered list comprehension

```
[ expression for name in list if filter ]
```

≈ համարժեք է

```
for element in list:  
    if filter:  
        expression
```

Նույնն է ինչ հասարակ list comprehension-ը, միայն **expression** –ը չի կիրառվում ամեն անդամի վրա

```
>>> lst = [3, 6, 2, 7, 1, 9]  
>>> [x * 2 for x in lst if x > 4]  
[12, 14, 18]
```

6, 7, 9 բաժարարում են $x > 4$ պայմանին

list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> lst = [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> lst = [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> vec = [-4, -2, 0, 2, 4]
>>> # create a new list with the values doubled of vec
```

list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> lst = [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> vec = [-4, -2, 0, 2, 4]
>>> # create a new list with the values doubled of vec
>>> [x*2 for x in vec]
[-8, -4, 0, 4, 8]
```

list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> lst = [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> vec = [-4, -2, 0, 2, 4]
>>> # create a new list with the values doubled of vec
>>> [x*2 for x in vec]
[-8, -4, 0, 4, 8]
>>> # filter the vec list : exclude negative numbers
```

list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> lst = [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> vec = [-4, -2, 0, 2, 4]
>>> # create a new list with the values doubled of vec
>>> [x*2 for x in vec]
[-8, -4, 0, 4, 8]
>>> # filter the vec list : exclude negative numbers
>>> [x for x in vec if x >= 0]
[0, 2, 4]
```

list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> lst = [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> vec = [-4, -2, 0, 2, 4]
>>> # create a new list with the values doubled of vec
>>> [x*2 for x in vec]
[-8, -4, 0, 4, 8]
>>> # filter the vec list : exclude negative numbers
>>> [x for x in vec if x >= 0]
[0, 2, 4]
>>> # apply function abs() to all the elements
```


list comprehension

```
>>> lst = []
>>> for x in [1,2,3]:
    for y in [3,1,4]:
        if x!=y:
            lst.append((x,y))

>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> lst = [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
>>> lst
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>> vec = [-4, -2, 0, 2, 4]
>>> # create a new list with the values doubled of vec
>>> [x*2 for x in vec]
[-8, -4, 0, 4, 8]
>>> # filter the vec list : exclude negative numbers
>>> [x for x in vec if x >= 0]
[0, 2, 4]
>>> # apply function abs() to all the elements
>>> [abs(x) for x in vec]
[4, 2, 0, 2, 4]
```

list comprehension

```
>>> # call a method on each element  
>>> fruits = ['banana', 'raspberry', 'grapefruit']
```

list comprehension

```
>>> # call a method on each element
>>> fruits = [' banana', ' raspberry ', ' grapefruit ']
>>> [fruit.strip() for fruit in fruits]
['banana', 'raspberry', 'grapefruit']
```

list comprehension

```
>>> # call a method on each element
>>> fruits = ['  banana', '  raspberry ', ' grapefruit  ']
>>> [fruit.strip() for fruit in fruits]
['banana', 'raspberry', 'grapefruit']

>>> # create a list of 2-tuples like (number, square)
```

list comprehension

```
>>> # call a method on each element
>>> fruits = [' banana', ' raspberry ', ' grapefruit ']
>>> [fruit.strip() for fruit in fruits]
['banana', 'raspberry', 'grapefruit']
```

```
>>> # create a list of 2-tuples like (number, square)
>>> [(x, x**2) for x in range(6)]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

list comprehension

```
>>> # call a method on each element
>>> fruits = ['  banana', '  raspberry ', ' grapefruit  ']
>>> [fruit.strip() for fruit in fruits]
['banana', 'raspberry', 'grapefruit']
```

```
>>> # create a list of 2-tuples like (number, square)
>>> [(x, x**2) for x in range(6)]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

```
>>> from math import pi
>>> [str(round(pi, i)) for i in range(1, 6)]
```

list comprehension

```
>>> # call a method on each element
>>> fruits = [' banana', ' raspberry ', ' grapefruit ']
>>> [fruit.strip() for fruit in fruits]
['banana', 'raspberry', 'grapefruit']
```

```
>>> # create a list of 2-tuples like (number, square)
>>> [(x, x**2) for x in range(6)]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

```
>>> from math import pi
>>> [str(round(pi, i)) for i in range(1, 6)]
['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

list comprehension

```
>>> # call a method on each element
>>> fruits = [' banana', ' raspberry ', ' grapefruit ']
>>> [fruit.strip() for fruit in fruits]
['banana', 'raspberry', 'grapefruit']
```

```
>>> # create a list of 2-tuples like (number, square)
>>> [(x, x**2) for x in range(6)]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

```
>>> from math import pi
>>> [str(round(pi, i)) for i in range(1, 6)]
['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

```
>>> # flatten a list using a listcomp with two 'for'
>>> vec = [[1,2,3], [4,5,6], [7,8,9]]
```


list comprehension

```
>>> # call a method on each element
>>> fruits = [' banana', ' raspberry ', ' grapefruit ']
>>> [fruit.strip() for fruit in fruits]
['banana', 'raspberry', 'grapefruit']
```

```
>>> # create a list of 2-tuples like (number, square)
>>> [(x, x**2) for x in range(6)]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

```
>>> from math import pi
>>> [str(round(pi, i)) for i in range(1, 6)]
['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

```
>>> # flatten a list using a listcomp with two 'for'
>>> vec = [[1,2,3], [4,5,6], [7,8,9]]
>>> [num for elem in vec for num in elem]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Exercise 1

Գրեք list comprehension որը:

հաշվում է 1-ից 100 գույգ integer թվերի գումարը

Exercise 1

Գրեք list comprehension նրբ:

հաշվում է 1-ից 100 գույգ integer թվերի գումարը

```
x = sum([i for i in range(1, 101) if not i % 2])
```

Exercise 1

Գրեք list comprehension նրբ:

հաշվում է 1-ից 100 գույգ integer թվերի գումարը

```
x = sum([i for i in range(1, 101) if not i % 2])
```

```
x = sum([i for i in range(0, 101, 2)])
```

Exercise 1

Գրեք list comprehension որը:

հաշվում է 1-ից 100 գույգ integer թվերի գումարը

```
x = sum([i for i in range(1, 101) if not i % 2])
```

```
x = sum([i for i in range(1, 101) if i % 2 == 0])
```

```
x = sum([i for i in range(0, 101, 2)])
```

Exercise 1

Գրեք list comprehension ուր:

հաշվում է 1-ից 100 գույգ integer թվերի գումարը

```
x = sum([i for i in range(1, 101) if not i % 2])
```

```
x = sum([i for i in range(1, 101) if i % 2 == 0])
```

```
x = sum([i for i in range(0, 101, 2)])
```

```
x = sum(i for i in range(0, 101, 2))
```

Character encoding/ կոդավորում

String (**str**) object-ը պարունակում է character-ների դասավորված հաջորդականություն:

- a b c ... z and A B C ... Z
- decimal digits: 0 1 2 3 4 5 6 7 8 9
- punctuation, operators and symbols:
'!"#\$%&\`()*+,-./:;<=>?@[\\]^_`{|}~'
- More later

Character encoding/ կոդավորում

String (**str**) object-ը պարունակում է character-ների դասավորված հաջորդականություն:

- a b c ... z and A B C ... Z
- decimal digits: 0 1 2 3 4 5 6 7 8 9
- punctuation, operators and symbols:
'!"#\$%&\`()'+,-./:;<=>?@[\\]^_`{|}~'
- More later

Ամեն character-ին համապատասխանեցվում է որոշակի bit-ային կոդ (**bit-encoding**)
հետո, այդ bit-ային կոդը քարտեզվում է հետ իր character-ին

Character encoding/ կոդավորում

String (**str**) object-ը պարունակում է character-ների դասավորված հաջորդականություն:

- a b c ... z and A B C ... Z
- decimal digits: 0 1 2 3 4 5 6 7 8 9
- punctuation, operators and symbols:
'!"#\$%&\`()'+,-./:;<=>?@[\\]^_`{|}~'
- More later

Ամեն character-ին համապատասխանեցվում է որոշակի bit-ային կոդ (**bit-encoding**)
հետո, այդ bit-ային կոդը քարտեզվում է հետ իր character-ին

Տարիներ շարունակ, անգլերեն տառերի կոդավորման ստանդարտը եղել է

American Standard Code for Information Interchange (ASCII)

ASCII

32		48	0	64	@	80	P	96	^	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(56	8	72	H	88	X	104	h	120	x
41)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	=	77	M	93]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o		

ASCII

32		48	0	64	@	80	P	96	^	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(56	8	72	H	88	X	104	h	120	x
41)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	=	77	M	93]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o		

The code for a is 97, which is 01100001 in binary or 0x61 in hexadecimal notation

ASCII

32		48	0	64	@	80	P	96	^	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(56	8	72	H	88	X	104	h	120	x
41)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	=	77	M	93]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o		

The code for a is 97, which is 01100001 in binary or 0x61 in hexadecimal notation

Ամեն ASCII character-ի կոդը տեղավորվում է
1 byte-ում (8 bits)

functions ord() and chr()

```
>>> ord('a')  
97  
>>> ord('?')  
63  
>>> ord('\n')  
10
```

Function **ord()** takes a character (i.e., a string of length 1) as input and returns its ASCII code

functions ord() and chr()

```
>>> ord('a')
97
>>> ord('?')
63
>>> ord('\n')
10
>>> chr(10)
'\n'
>>> chr(63)
'?'
>>> chr(97)
'a'
```

Function **ord()** takes a character (i.e., a string of length 1) as input and returns its ASCII code

Function **chr()** takes an ASCII encoding (i.e., a non-negative integer) and returns the corresponding character

functions ord() and chr()

```
>>> ord('a')
97
>>> ord('?')
63
>>> ord('\n')
10
>>> chr(10)
'\n'
>>> chr(63)
'?'
>>> chr(97)
'a'
>>> ord('u')
1407
>>> chr(1407)
'u'
```

Function **ord()** takes a character (i.e., a string of length 1) as input and returns its ASCII code

Function **chr()** takes an ASCII encoding (i.e., a non-negative integer) and returns the corresponding character



1407?

Beyond ASCII

A string (**str**) object-ը պարունակում է character-ների դասավորված հաջորդականություն:

- a b c ... z and A B C ... Z
- decimal digits: 0 1 2 3 4 5 6 7 8 9
- punctuation, operators and symbols:
'!"#\$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
- character-ներ մյուս լեզուներից
- symbol-ներ math-ից, science-ից, engineering-ից,...

Beyond ASCII

A string (**str**) object-ը պարունակում է character-ների դասավորված հաջորդականություն:

- a b c ... z and A B C ... Z
- decimal digits: 0 1 2 3 4 5 6 7 8 9
- punctuation, operators and symbols:
'! "\$ % & \ ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~'
- character-ներ մյուս լեզուներից
- symbol-ներ math-ից, science-ից, engineering-ից,...

ASCII-ում միայն 128 character կա

Beyond ASCII

A string (**str**) object-ը պարունակում է character-ների դասավորված հաջորդականություն:

- a b c ... z and A B C ... Z
- decimal digits: 0 1 2 3 4 5 6 7 8 9
- punctuation, operators and symbols:
'! "\$ % & \ ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~'
- character-ներ մյուս լեզուներից
- symbol-ներ math-ից, science-ից, engineering-ից,...

ASCII-ում միայն 128 character կա

Unicode-ն է ստեղծվում, որը կլինի համընդհանուր character-ների կոդավորման սխեման

Unicode

Unicode-ում, ամեն character-ը նկարագրվում է integer կոդային կետով (integer code point).

code point-ը տվյալ character-ի id-ն է

```
>>> '\u0061'  
'a'  
>>> '\u0064\u0061d'  
'dad'
```

Unicode

Unicode-ում, ամեն character-ը նկարագրվում է integer կոդային կետով (integer code point).

code point-ը տվյալ character-ի id-ն է

\u start of Unicode
code point

```
>>> '\u0061'  
'a'  
>>> '\u0064\u0061d'  
'dad'
```

Unicode

Unicode-ում, ամեն character-ը նկարագրվում է integer կոդային կետով (integer code point).

code point-ը տվյալ character-ի id-ն է

a-ի code point-ը integer է 0x0061 hex արժեքով

- ASCII-ն Unicode-ի ենթաբազմություն է

\u start of Unicode
code point

```
>>> '\u0061'  
'a'  
>>> '\u0064\u0061d'  
'dad'
```

Unicode

Unicode-ում, ամեն character-ը նկարագրվում է integer կոդային կետով (integer code point).

code point-ը տվյալ character-ի id-ն է

a-ի code point-ը integer է 0x0061 hex արժեքով

- ASCII-ն Unicode-ի ենթաբազմություն է

Unicode-ով կարելի է գրել

- english
- cyrillic
- ...
- [հայերեն](#)

\u start of Unicode
code point

```
>>> '\u0061'
'a'
>>> '\u0064\u0061d'
'dad'
>>> '\u0409\u0443\u0431\u043e\u04
3c\u0438\u0440'
'Љубомир'
>>> '\u0531'
'Ա'
>>> '\u0531\u0532\u0533\u0534'
'ԱԲԳԴ'
```

Comparing Unicode

Unicode code point-ները, լինելով integer, ունեն
բնական դասավորվածություն

Comparing Unicode

Unicode code point-ները, լինելով integer, ունեն
բնական դասավորվածություն

Unicode-ի դիզայնը հետևյալն է

Եթե character-ը տվյալ լեզվի
այբուբենում գտնվում է մյուս
տառից առաջ, ապա նրա code-
point-ը փոքր է հաջորդինից

Comparing Unicode

Unicode code point-ները, լինելով integer, ունեն
բնական դասավորվածություն

Unicode-ի դիզայնը հետևյալն է

Եթե character-ը տվյալ լեզվի
այբուբենում գտնվում է մյուս
տառից առաջ, ապա նրա code-
point-ը փոքր է հաջորդինից

```
>>> s1 = '\u0021'  
>>> s1  
'!'  
>>> s2 = '\u0409'  
>>> s2  
'ђ'  
>>> s1 < s2  
True  
>>> '\u0531\u0532'  
'ԱԲ'
```

Unicode Transformation Format (UTF)

Unicode Transformation Format (UTF)

Unicode string-ը code point-ների
հաջորդականություն է, որոնք integer-ներ են 0-ից
0x10FFFF տիրույթում

Unicode Transformation Format (UTF)

Unicode string-ը code point-ների
հաջորդականություն է, որոնք integer-ներ են 0-ից
0x10FFFF տիրույթում

Ի տարբերություն ASCII-ի, Unicode code point-ը չէ,
որ պահվում է հիշողության մեջ

Unicode Transformation Format (UTF)

Unicode string-ը code point-ների հաջորդականություն է, որոնք integer-ներ են 0-ից 0x10FFFF տիրույթում

Ի տարբերություն ASCII-ի, Unicode code point-ը չէ, որ պահվում է հիշողության մեջ

կանոնը որով Unicode character-ը (code point-ը) թարգմանվում է byte-երի հաջորդականության կոչվում է **encoding** կոդավորում.

Unicode Transformation Format (UTF)

UTF stands for Unicode Transformation Format.

ኃሰኑ ሰከ ምሰከ Unicode encodings:

UTF-8, UTF-16, and UTF-32.

Unicode Transformation Format (UTF)

UTF stands for Unicode Transformation Format.

Կան մի քանի Unicode encodings:

UTF-8, UTF-16, and UTF-32.

UTF-8 նախընտրելի encoding-ն է e-mail և web page-երի

Python **3**-ի default encoding-ը **UTF-8** է

Unicode Transformation Format (UTF)

UTF stands for Unicode Transformation Format.

Կան մի քանի Unicode encodings:
UTF-8, UTF-16, and UTF-32.

UTF-8 նախընտրելի encoding-ն է e-mail և web page-երի

Python **3**-ի default encoding-ը **UTF-8** է

UTF-8 -ով, ամեն ASCII character ունի նույն ASCII 8-bit encoding.

Armenian Unicode range

```
start = 1329 # 0x0531 armenian chapital a  
end   = 1423 # 0x058f armenian dram sign
```

```
for i in range(start, end + 1):  
    print(format(i, 'X'), end=' ') # hex  
    print(i, end=' ')             # dec  
    print(chr(i))                 # letter
```

531	1329	Ա
532	1330	Բ
533	1331	Գ
534	1332	Դ
535	1333	Ե
536	1334	Զ
537	1335	Է

...

Decoding bytes

Երբ file-ը internet-ից download է արվում, այն չունի encoding

- file-ը կարող է լինել picture, executable program, i.e. not a text file
- download արված file-ի պարունակությունը byte-երի sequence է, i.e. of **type bytes**

Decoding bytes

byte-երի method **decode()** վերցնում է կոդավորման անունը և կիրառում այն byte sequence-ի վրա

➤ default is UTF-8

```
>>> content
```

```
b'This is a text document\nposted on the\nWWW.\n'
```

```
>>> type(content)
```

```
<class 'bytes'>
```

Decoding bytes

byte-երի method **decode()** վերցնում է կոդավորման անունը և կիրառում այն byte sequence-ի վրա

➤ default is UTF-8

```
>>> content
b'This is a text document\nposted on the\nWWW.\n'
>>> type(content)
<class 'bytes'>
>>> s = content.decode('utf-8')
>>> type(s)
<class 'str'>
>>> s
'This is a text document\nposted on the\nWWW.\n'
```

Decoding bytes

byte-երի method **decode()** վերցնում է կոդավորման անունը և կիրառում այն byte sequence-ի վրա

➤ default is UTF-8

```
>>> content
b'This is a text document\nposted on the\nWWW.\n'
>>> type(content)
<class 'bytes'>
>>> s = content.decode('utf-8')
>>> type(s)
<class 'str'>
>>> s
'This is a text document\nposted on the\nWWW.\n'
>>> s = content.decode()
>>> s
'This is a text document\nposted on the\nWWW.\n'
>>>
```

Exercise 2

Գրեք ծրագիր որը:

- Կանչում է url (www.news.am/arm/)
- հաշվում է բոլոր character-ների թիվը այդ url-ում
- որոնք ընկած են [1329,1423] միջև

number of armenian characters: **23539**

Exercise 2

Գրեք ծրագիր որը:

- Կանչում է url (www.news.am/arm/)
- հաշվում է բոլոր character-ների թիվը այդ url-ում
- որոնք ընկած են [1329,1423] միջև

number of armenian characters: 23539

HINT: `pip install requests`

```
import requests
response = requests.get(url)
text = response.text
```

Solution

```
import requests
```

```
url = 'http://news.am/arm'
```

```
res = requests.get(url)
```

```
chars = [char for char in res.text  
          if ord(char) in range(1329,1423+1)]  
print("number of armenian characters:", len(chars))
```


Exercise 3

Գրեք function **armunicode()** որը:

- վերցնում է url (www.news.am/arm/) որպես input
- հաշվում է բոլոր character-ների թիվը այդ url-ում
- որոնք ընկած են [1329,1423] միջև

```
armunicode('http://news.am/arm/')
```

number of armenian characters: **23539**

Solution

```
import requests

def armunicode(url):
    res = requests.get(url)
    chars = [char for char in res.text
              if ord(char) in range(1329,1423+1)]

    print("number of armenian characters:",len(chars))

armunicode('http://google.am')
# number of armenian characters: 95
```

References

1. Downey, Allen B. *Think Python*. 1 edition. Sebastopol, CA: O'Reilly Media, 2012.
2. “Unicode.” *Wikipedia*, February 22, 2017.
<https://en.wikipedia.org/w/index.php?title=Unicode&oldid=766888752>.
3. “Requests: HTTP for Humans — Requests 2.13.0 Documentation.” Accessed February 24, 2017.
<http://docs.python-requests.org/en/master/>.