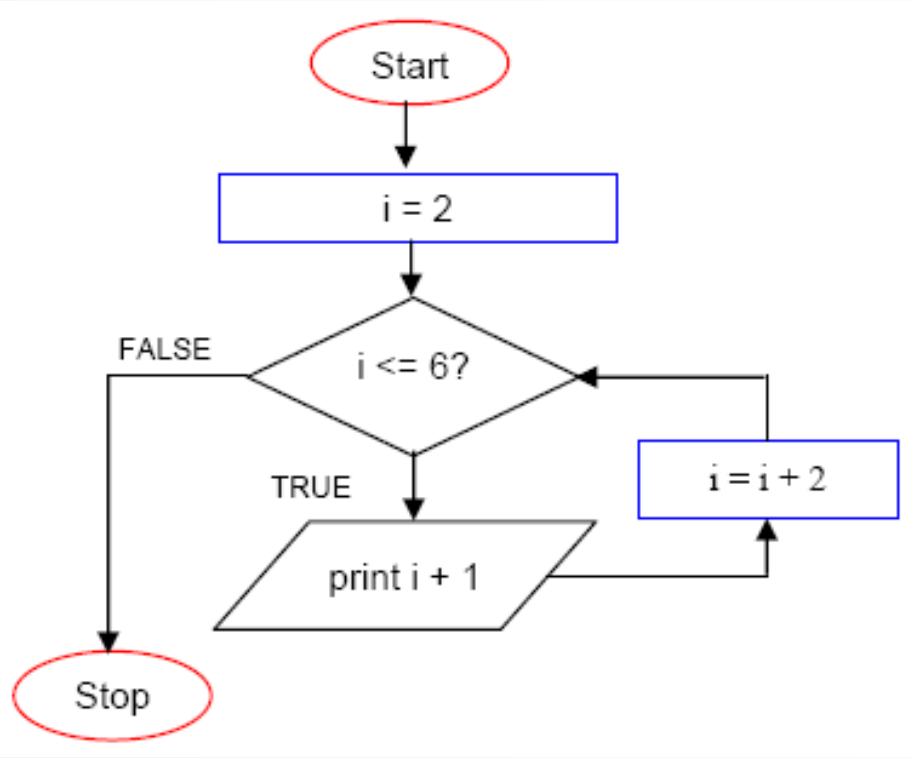
Loops

- Loops
  - For-loop, while-loop
- range()
- Iterating through the numbers, string, and list
- Nested loop
- Conditional statement
  - If/elif/else
- comparison and logical operators

NAZARBAYEV UNIVERSITY

Loops

- It is rarely useful that a computer performs each operation only once.
- The purpose of **loops** is to repeat the same, or similar, code a number of times.



for-loop

- The most common type is a for-loop.
- It executes some part of the code for predetermined number of times.

1. start with the keyword "`for`",
2. followed by the name of the variable that will be assigned

   all the values through which we want to loop
3. then the keyword "`in`",
4. then a list or something that acts like it
5. then a colon "`:`".

```
for varaible in items:
    codes (loop body)
```

NAZARBAYEV UNIVERSITY

for-loop

- The most common type is a for-loop.
- It executes some part of the code for predetermined number of times.

1. start with the keyword "`for`",
2. followed by the name of the variable that will be assigned

   all the values through which we want to loop
3. then the keyword "`in`",
4. then a list or something that acts like it
5. then a colon "`:`".

```
for i in [0, 1, 2, 3, 4]:
    print(“hello, world!!”)
    print(i)
```

NAZARBAYEV UNIVERSITY

for-loop

- The most common type is a for-loop.
- It executes some part of the code for predetermined number of times.

1. start with the keyword "`for`",
2. followed by the name of the variable that will be assigned

   all the values through which we want to loop
3. then the keyword "`in`",
4. then a list or something that acts like it
5. then a colon "`:`".

```
for i in [‘a’, ‘b’, ‘c’, ‘d’, ‘e’]:
    print(“hello, world!!”)
    print(i)
```

NAZARBAYEV UNIVERSITY

range()

- We often use the for-loop together with the **range** function
- It pretends to return a list of numbers (it returns something more complex, we can consider it as list for now).

- `range(n)` -- numbers $0, 1, \ldots, n-1$; this is equivalent to `range(0,n)`;
- `range(m, n)` -- numbers $m, m+1, \ldots, n-1$;
- `range(m, n, s)` -- numbers $m, m+s, \ldots, m+sk$, where $k \in \mathbb{N}$ such that $m+sk < n \leq m+s(k+1)$.
  In other words, numbers from m to $n-1$ with step s, but we might not hit $n-1$, depending on the value of step s.

```
>>> list(range(5))
[0, 1, 2, 3, 4]
```

NAZARBAYEV UNIVERSITY

range()

```
>>> list(range(2))
[0, 1]
>>> list(range(1, 11))

>>> list(range(1, 2))

>>> list(range(2, 5, 1))

>>> list(range(1, 11, 2))

>>> list(range(5, 0, -1))
```

range()

```
>>> list(range(2))
[0, 1]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(1, 2))

>>> list(range(2, 5, 1))

>>> list(range(1, 11, 2))

>>> list(range(5, 0, -1))
```

range()

```
>>> list(range(2))
[0, 1]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(1, 2))
[1]
>>> list(range(2, 5, 1))

>>> list(range(1, 11, 2))

>>> list(range(5, 0, -1))
```

range()

```
>>> list(range(2))
[0, 1]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(1, 2))
[1]
>>> list(range(2, 5, 1))
[2, 3, 4]
>>> list(range(1, 11, 2))

>>> list(range(5, 0, -1))
```

range()

```
>>> list(range(2))
[0, 1]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(1, 2))
[1]
>>> list(range(2, 5, 1))
[2, 3, 4]
>>> list(range(1, 11, 2))
[1, 3, 5, 7, 9]
>>> list(range(5, 0, -1))
```

range()

```
>>> list(range(2))
[0, 1]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(1, 2))
[1]
>>> list(range(2, 5, 1))
[2, 3, 4]
>>> list(range(1, 11, 2))
[1, 3, 5, 7, 9]
>>> list(range(5, 0, -1))
[5, 4, 3, 2, 1]
```

for-loop

```
for i in [0, 1, 2, 3, 4]:
    print("hello, world!!")
```

```
a = [0, 1, 2, 3, 4]
for i in a:
    print("hello, world!!")
```

```
for i in range(5):
    print("hello, world!!")
```

Output:

```
Hello, world!!
Hello, world!!
Hello, world!!
Hello, world!!
Hello, world!!
```

for-loop

```
for i in [0, 1, 2, 3, 4]:
    print(i)
```

```
a = [0, 1, 2, 3, 4]
for i in a:
    print(i)
```

```
for i in range(5):
    print(i)
```

Output:

```
0
1
2
3
4
```

NAZARBAYEV UNIVERSITY

for-loop

- Here, we don't need the variable i.
- We can use the underscore '_' instead:

```
for _ in [0, 1, 2, 3, 4]:
    print("hello, world!!")
```

```
a = [0, 1, 2, 3, 4]
for _ in a:
    print("hello, world!!")
```

```
for _ in range(5):
    print("hello, world!!")
```

for-loop

- Here, we don't need the variable i.
- We can use the underscore '_' instead:

```
for _ in [0, 1, 2, 3, 4]:
    print("hello, world!!")
```

```
a = [0, 1, 2, 3, 4]
for _ in a:
    print("hello, world!!")
```

```
for _ in range(5):
    print("hello, world!!")
```

Iterating through the numbers (list)

```
Here = 20
There = 24
for k in range(Here, There):
    print(k)
```

Output:

```
20
21
22
23
```

NAZARBAYEV UNIVERSITY

Iterating through a string

```
s = 'abcde'
for c in s:
    print(c)
```

Output:

```
a
b
c
d
e
```

In this example, the ‘for-loop’ variable is c.
One at a time, it takes on the value of each character in s

Iterating through a list

```
s = ['kor', 'kz', 'USA', 'jp']
for c in s:
    print(c)
```

Output:

```
kor
kz
USA
jp
```

In this example, the ‘for-loop’ variable is c.
One at a time, it takes on the value of each character in s

Iterating in for-loop

- **We are repeating the codes in for-loop 4 times**

```
n = 4
s = 0
for k in range(n):
    x = 2**k
    s = s+x
print(s)
```

Output:

NAZARBAYEV UNIVERSITY

Iterating in for-loop

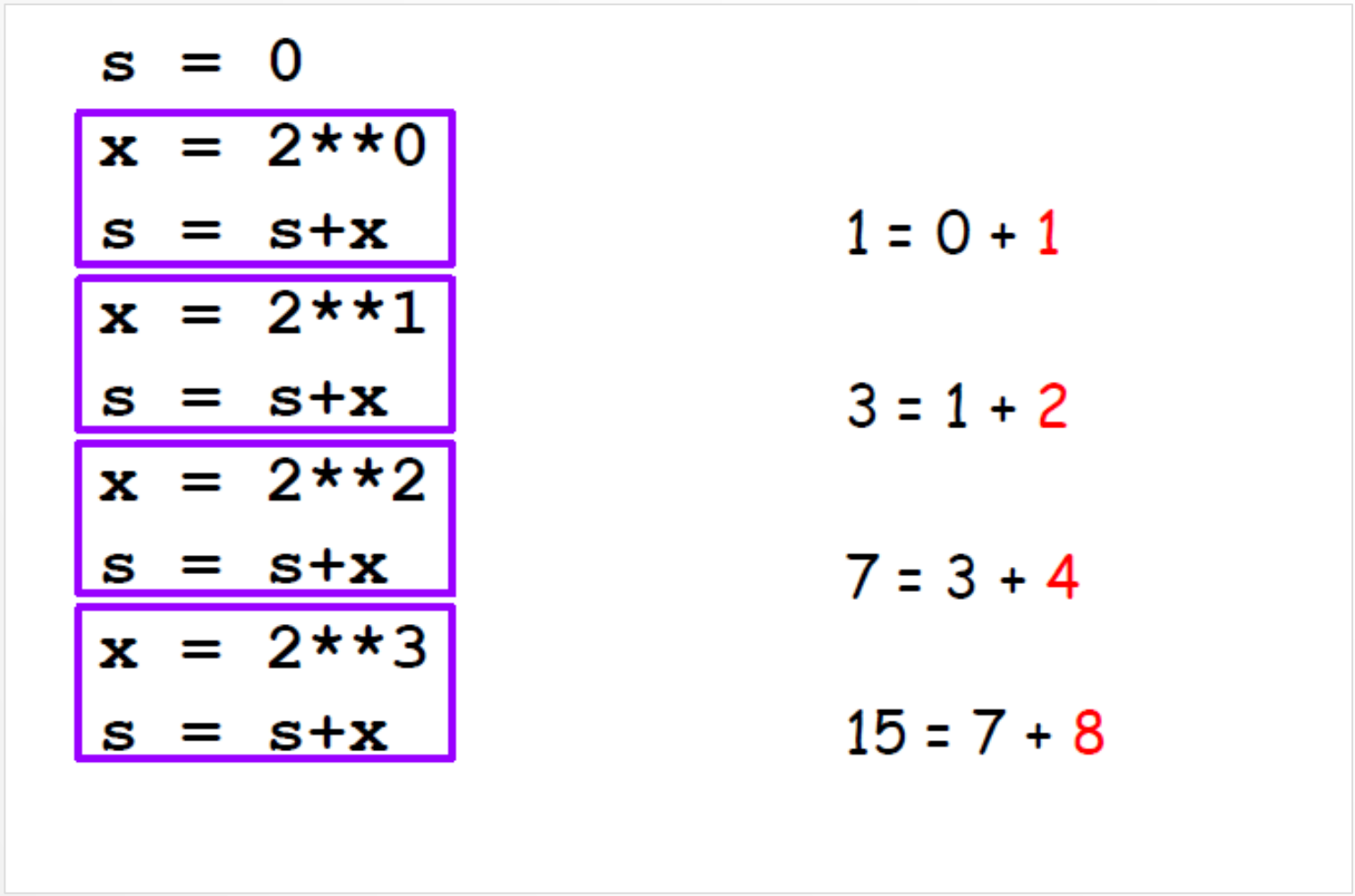
- **We are repeating the codes in for-loop 4 times**

```
n = 4
s = 0
for k in range(n):
    x = 2**k
    s = s+x
print(s)
```

Output:

15

Iterating in for-loop



NAZARBAYEV UNIVERSITY

Nested loop

- The placing of one loop inside the body of another loop is called **nesting**.
- When working with nested loops, the outer loop changes only after the inner loop is completely finished.

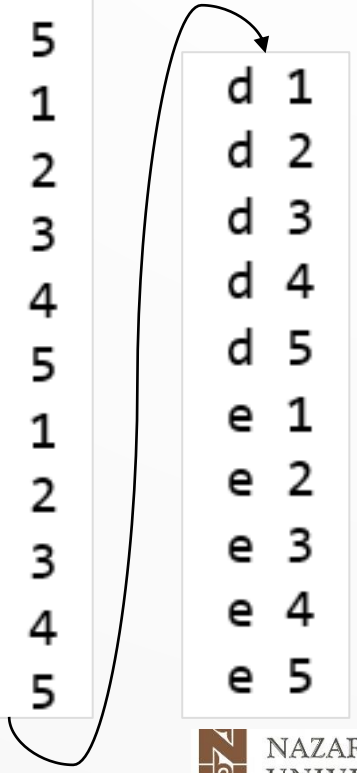
```
s1 = ['a', 'b', 'c', 'd', 'e']
n1 = [1, 2, 3, 4, 5]

for i in s1:
    for j in n1:
        print(i, j)
```

Outer

inner

Output:

```
a 1
a 2
a 3
a 4
a 5
b 1
b 2
b 3
b 4
b 5
c 1
c 2
c 3
c 4
c 5
d 1
d 2
d 3
d 4
d 5
e 1
e 2
e 3
e 4
e 5
```



NAZARBAYEV UNIVERSITY

Iterating through a list

```
s = ['kor', 'kz', 'USA', 'jp']
```

1) *Using nested loop*
2) *Using .join() function*

Expected Output:

```
k
o
r
k
z
U
S
A
j
p
```

NAZARBAYEV UNIVERSITY

Iterating through a list

```
s = ['kor', 'kz', 'USA', 'jp']
for c in s:
    for j in range(len(c)):
        print(c[ j ])
```

Output:

```
k
o
r
k
z
U
S
A
j
p
```

NAZARBAYEV UNIVERSITY

Iterating through a list

```
s = ['kor', 'kz', 'USA', 'jp']
s2 = ‘’.join(s)
for c in range(len(s2)):
    print(s2[ c ])
```

Output:

```
k
o
r
k
z
U
S
A
j
p
```

NAZARBAYEV UNIVERSITY

Left-shifting a string

- Write the codes to reverse the string ‘a’

```
a = 'abcdef'

...
```

Expected Output:

```
fedcba
```

Left-shifting a string

- All letters are concatenated in opposite direction

```
a = 'abcdef'
t = ''
for i in a:
    t = i + t
    print(t)
```

Output:

```
a
ba
cba
dcba
edcba
fedcba
```

NAZARBAYEV UNIVERSITY

Left-shifting a string

- It reads the letters from the end-offset to the first-offset

```
a = 'abcdef'

t2 = '';

for i in range(len(a)-1, -1, -1):

    t2 = t2 + a[i]

    print(t2)
```

Output:

```
f
fe
fed
fedc
fedcb
fedcba
```

Conditionals (if, elif, else)

- If some condition is true, do this.
- If condition1 is true, do job1, else if condition2 is true, do j ob2, else do job3

```
n = int(input("Input an integer: "))
if n < 0:
    print("Number", n, "is negative.")
elif n > 0:
    print("Number", n, "is positive.")
else:
    print("Number", n, "has an identity crisis!")
```

Comparison and logical operators

Operator	Example	Meaning
<	a < b	The value of a is smaller than the value of b
<=	a <= b	The value of a is smaller than or equal to the value of b
>	a > b	The value of a is bigger than the value of b
>=	a >= b	The value of a is bigger than or equal to the value of b
==	a == b	The value of a is equal to the value of b (but not necessarily identical!)
!=	a != b	The value of a is not equal to the value of b
is	a is b	a and b are exactly the same object
is not	a is not b	a and b are not exactly the same object
or	a or b	a is true or b is true (or both); a and b can be non-Boolean values
and	a and b	Both a and b are true; a and b can be non-Boolean values
not	not a	True if a is false; False otherwise; a can be a non-Boolean value

Comparing sequences Function ord()

- **Lexicographically**

```
>>> (1, 2, 3) < (1, 2, 4)   # True
>>> [1, 2, 3] < [1, 2, 4]   # True
>>>  'ABC' > 'C'             # False
>>> 'Pascal' < 'Python'      # True
>>> (1, 2, 3, 4) < (1, 2, 4)  # True
>>> (1, 2) < (1, 2, 1)        # True
>>> (1, 2, 3) == (1.0, 2.0, 3.0) # True
>>> (1, 2, ('aa', 'ab')) < (1, 2, ('abc ', 'a'))   # True
```

EV UNIVERSITY

ASCII table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

NAZARBAYEV UNIVERSITY