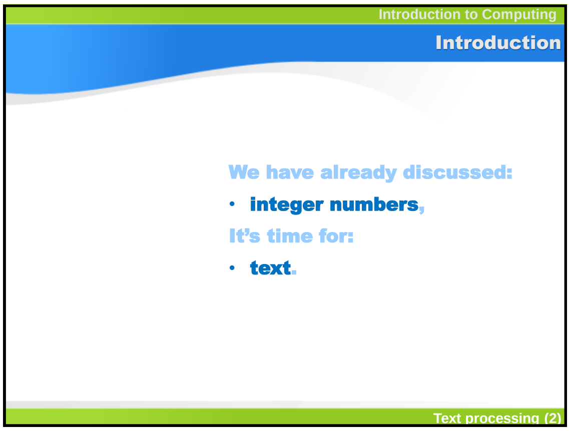
Introduction to Computing

Jerzy Nawrocki

Faculty of Computing & Telecom.
Poznan University of Technology
jerzy.nawrocki@put.poznan.pl

Text processing

https://ultimatehistoryproject.com/the-medieval-scribe.html



Introduction to Computing

Introduction

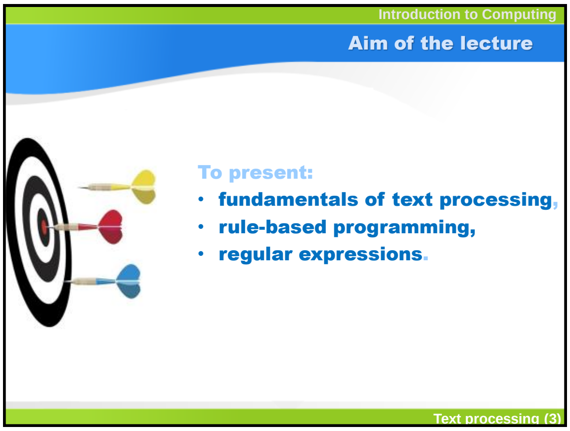
We have already discussed:

- **integer numbers**,

It's time for:

- **text**.

Text processing (2)



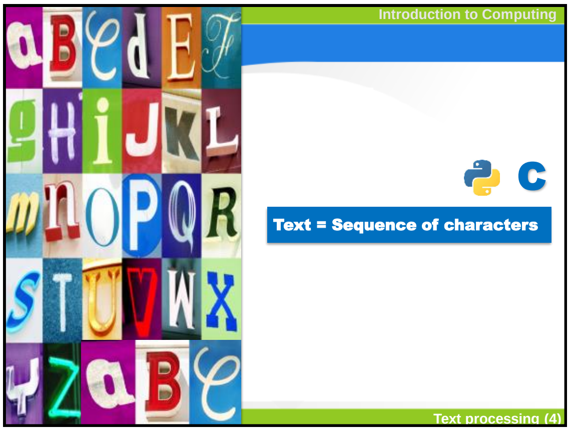
Introduction to Computing

Aim of the lecture

To present:

- **fundamentals of text processing**,
- **rule-based programming**,
- **regular expressions**.

Text processing (3)

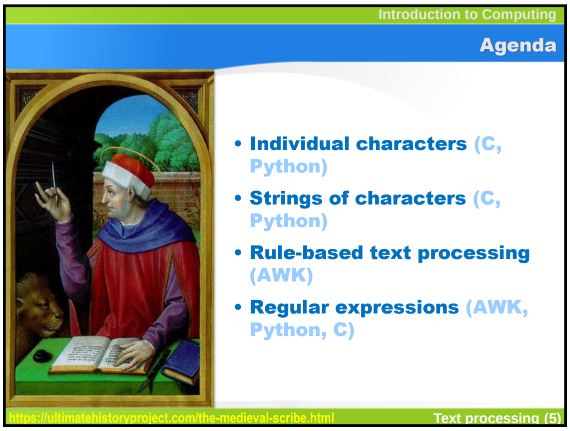


Introduction to Computing

C

Text = Sequence of characters

Text processing (4)



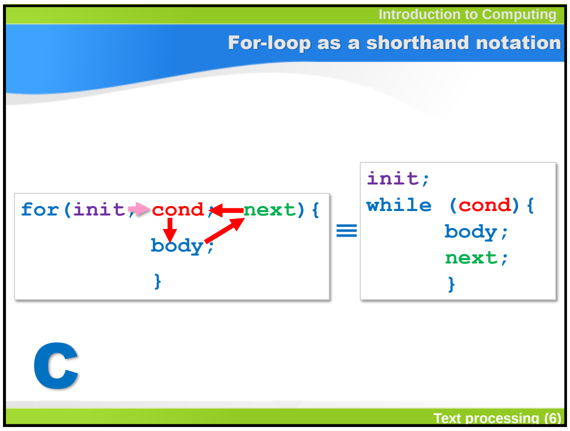
Introduction to Computing

Agenda

- **Individual characters** (C, Python)
- **Strings of characters** (C, Python)
- **Rule-based text processing** (AWK)
- **Regular expressions** (AWK, Python, C)

https://ultimatehistoryproject.com/the-medieval-scribe.html

Text processing (5)



Introduction to Computing

For-loop as a shorthand notation

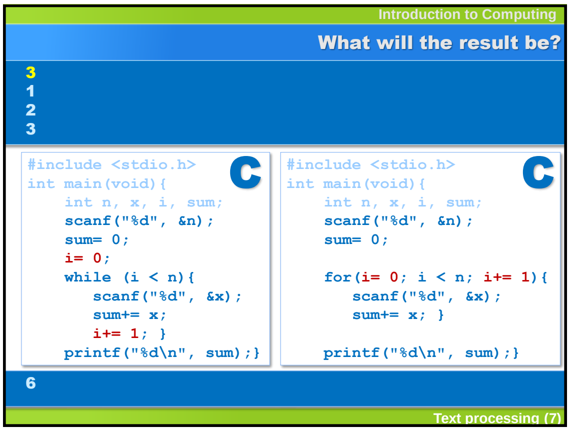
```
for(init; cond ; next){
          body;
          }
```

≡

```
init;
while (cond){
        body;
        next;
     }
```

C

Text processing (6)



Introduction to Computing

What will the result be?

```
3
1
2
3
```

```
#include <stdio.h>
int main(void){
    int n, x, i, sum;
    scanf("%d", &n);
    sum= 0;
    i= 0;
    while (i < n){
       scanf("%d", &x);
       sum+= x;
       i+= 1; }
    printf("%d\n", sum);}
```

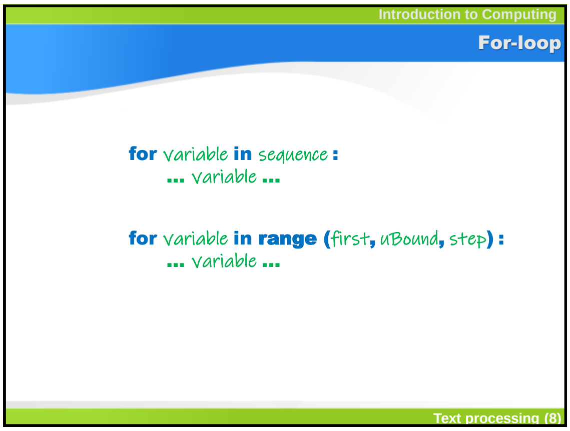
C

```
#include <stdio.h>
int main(void){
    int n, x, i, sum;
    scanf("%d", &n);
    sum= 0;

    for(i= 0; i < n; i+= 1){
       scanf("%d", &x);
       sum+= x; }

    printf("%d\n", sum);}
```

C

```
6
```

Text processing (7)

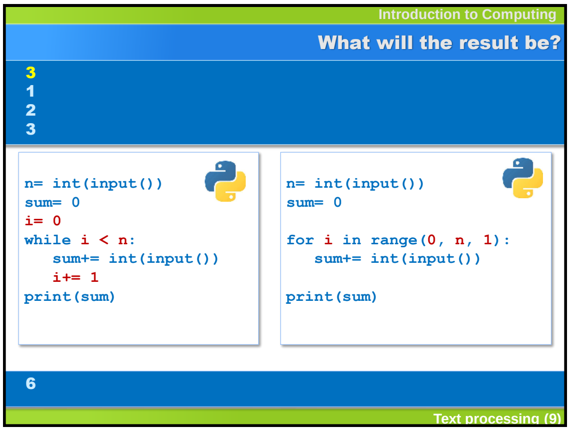


Introduction to Computing

For-loop

```
for variable in sequence :
    ... variable ...

for variable in range (first, uBound, step) :
    ... variable ...
```

Text processing (8)



Introduction to Computing

What will the result be?

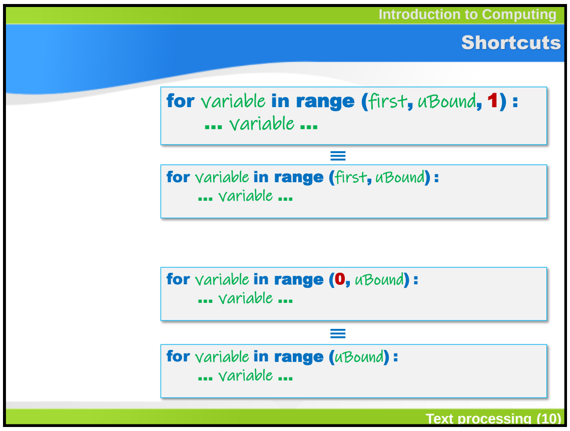
```
3
1
2
3
```

```
n= int(input())
sum= 0
i= 0
while i < n:
    sum+= int(input())
    i+= 1
print(sum)
```

```
n= int(input())
sum= 0

for i in range(0, n, 1):
    sum+= int(input())

print(sum)
```

```
6
```

Text processing (9)



Introduction to Computing

Shortcuts

```
for variable in range (first, uBound, 1) :
    ... variable ...
```

≡

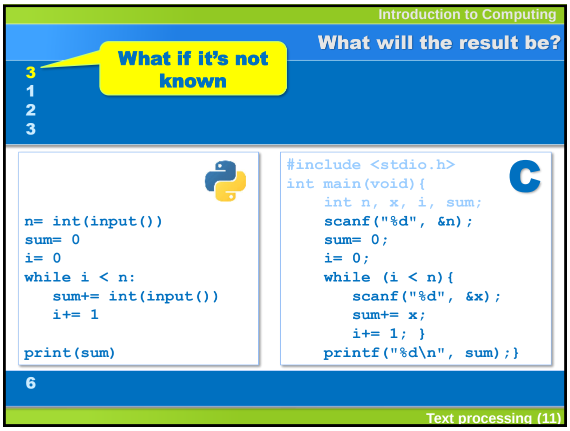
```
for variable in range (first, uBound) :
    ... variable ...
```

```
for variable in range (0, uBound) :
    ... variable ...
```

≡

```
for variable in range (uBound) :
    ... variable ...
```

Text processing (10)



Introduction to Computing

What will the result be?

What if it's not known

```
3
1
2
3
```

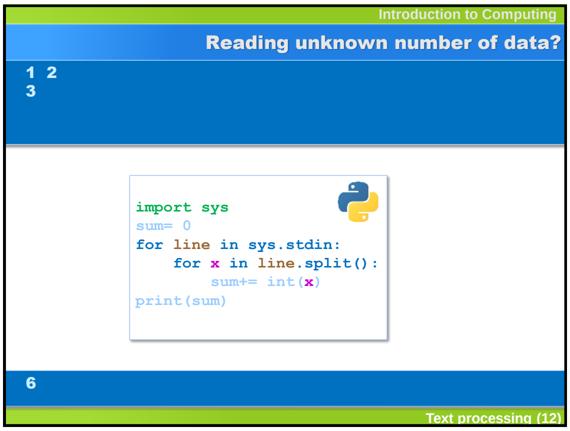
```
n= int(input())
sum= 0
i= 0
while i < n:
    sum+= int(input())
    i+= 1

print(sum)
```

```
#include <stdio.h>
int main(void){
    int n, x, i, sum;
    scanf("%d", &n);
    sum= 0;
    i= 0;
    while (i < n){
       scanf("%d", &x);
       sum+= x;
       i+= 1; }
    printf("%d\n", sum);}
```

C

```
6
```

Text processing (11)



Introduction to Computing

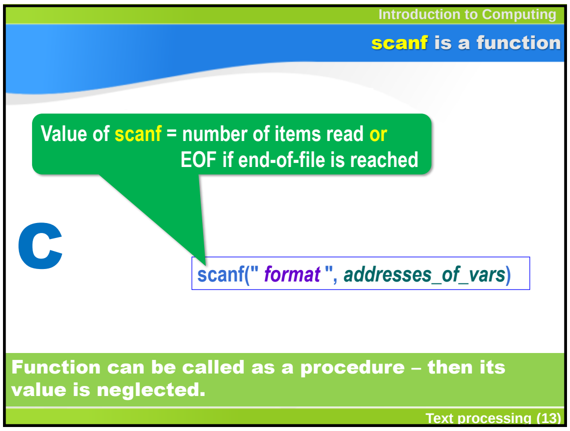
Reading unknown number of data?

```
1 2
3
```

```
import sys
sum= 0
for line in sys.stdin:
    for x in line.split():
        sum+= int(x)
print(sum)
```

```
6
```

Text processing (12)

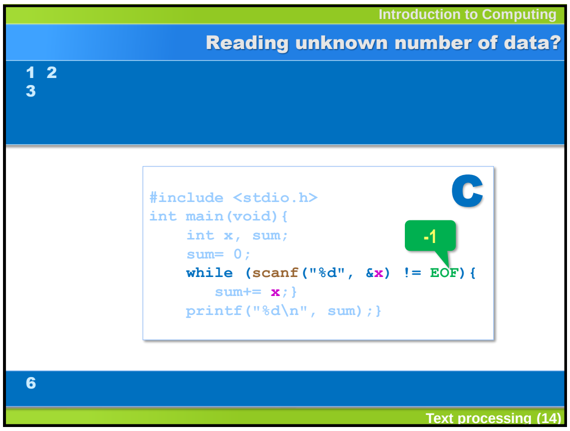


scanf is a function

Value of scanf = number of items read or
EOF if end-of-file is reached

scanf(" *format* ", *addresses_of_vars*)

Function can be called as a procedure – then its value is neglected.



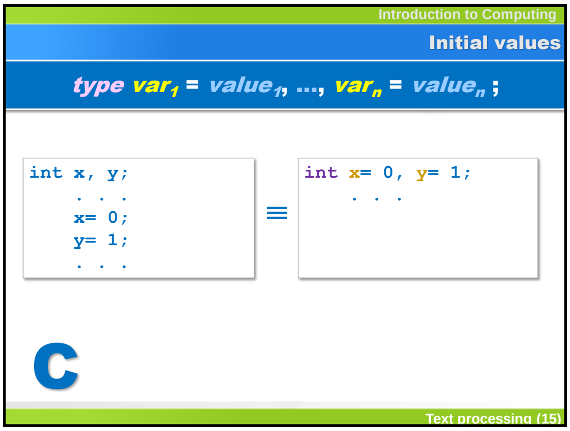
Reading unknown number of data?

```
1 2
3
```

```
#include <stdio.h>
int main(void){
    int x, sum;
    sum= 0;
    while (scanf("%d", &x) != EOF){
       sum+= x;}
    printf("%d\n", sum);}
```

-1

```
6
```



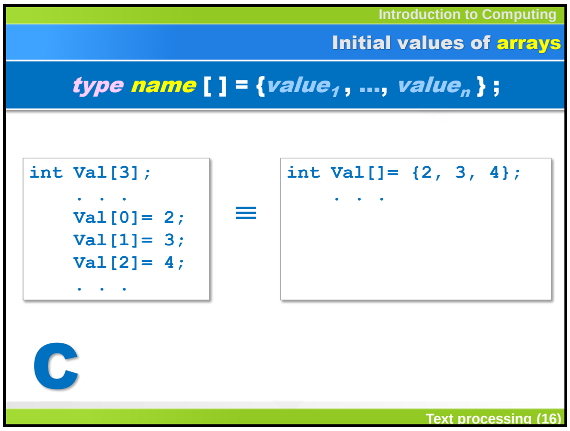
Initial values

type var$_1$ **=** ***value***$_1$**, …,** ***var***$_n$ **=** ***value***$_n$ **;**

```
int x, y;
    . . .
    x= 0;
    y= 1;
    . . .
```

≡

```
int x= 0, y= 1;
    . . .
```



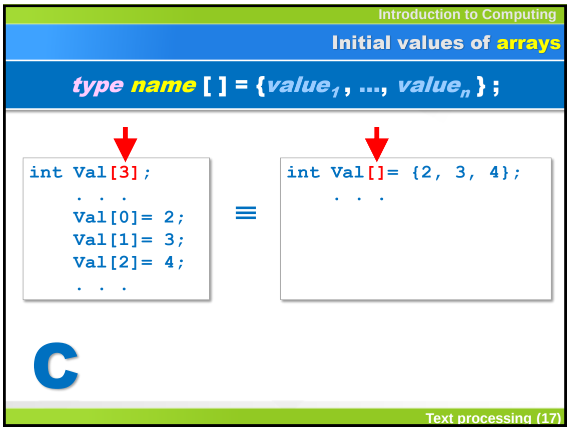
Initial values of arrays

type name **[ ] = {*****value***$_1$ **, …,** ***value***$_n$ **} ;**

```
int Val[3];
    . . .
    Val[0]= 2;
    Val[1]= 3;
    Val[2]= 4;
    . . .
```

≡

```
int Val[]= {2, 3, 4};
    . . .
```



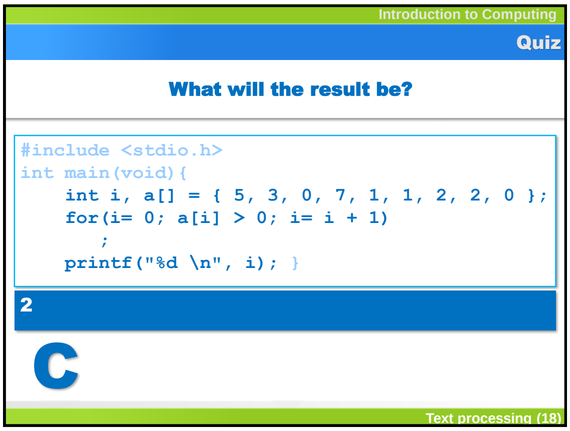
Initial values of arrays

type name **[ ] = {*****value***$_1$ **, …,** ***value***$_n$ **} ;**

```
int Val[3];
    . . .
    Val[0]= 2;
    Val[1]= 3;
    Val[2]= 4;
    . . .
```

≡

```
int Val[]= {2, 3, 4};
    . . .
```



Quiz

What will the result be?

```
#include <stdio.h>
int main(void){
    int i, a[] = { 5, 3, 0, 7, 1, 1, 2, 2, 0 };
    for(i= 0; a[i] > 0; i= i + 1)
       ;
    printf("%d \n", i); }
```

```
2
```

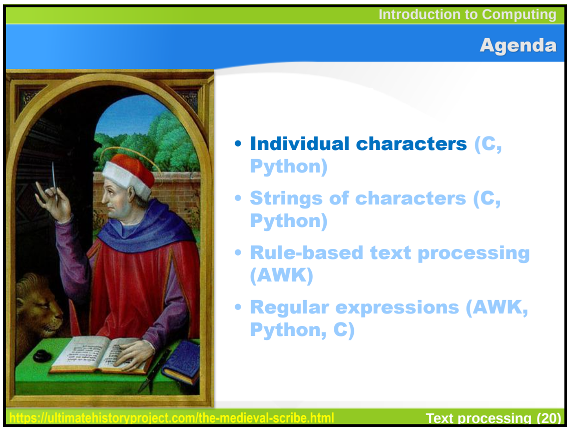
Introduction to Computing

Jerzy Nawrocki

Faculty of Computing & Telecom.
Poznan University of Technology
jerzy.nawrocki@put.poznan.pl

Text processing

https://ultimatehistoryproject.com/the-medieval-scribe.html



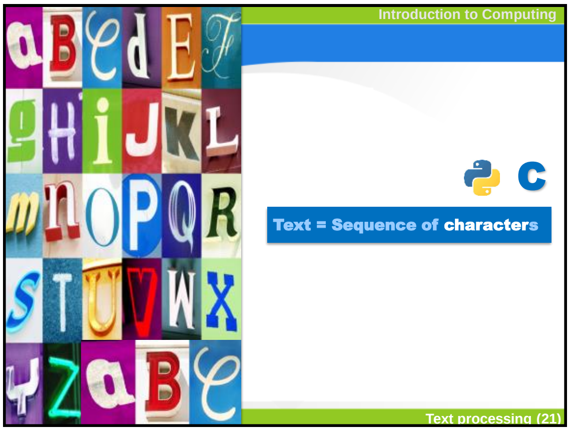
Introduction to Computing

Agenda

- **Individual characters** (C, Python)
- Strings of characters (C, Python)
- Rule-based text processing (AWK)
- Regular expressions (AWK, Python, C)

https://ultimatehistoryproject.com/the-medieval-scribe.html

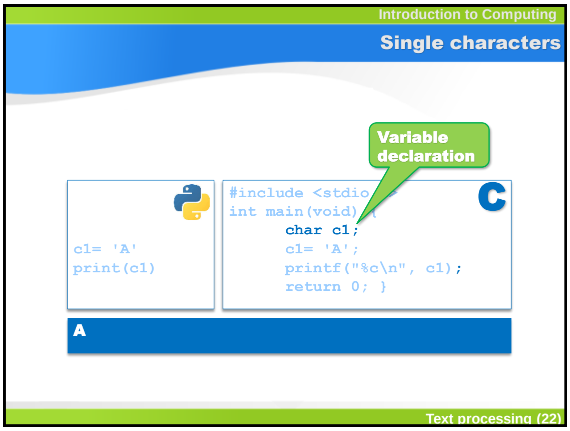
Text processing (20)



Introduction to Computing

Text = Sequence of characters

Text processing (21)



Introduction to Computing

Single characters

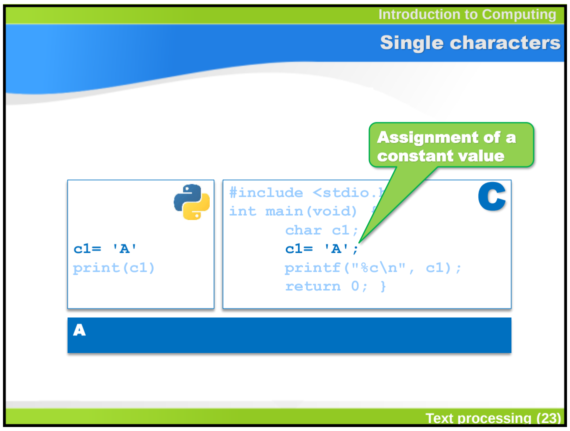
Variable declaration

```
c1= 'A'
print(c1)
```

```
#include <stdio
int main(void)
      char c1;
      c1= 'A';
      printf("%c\n", c1);
      return 0; }
```

```
A
```

Text processing (22)



Introduction to Computing

Single characters

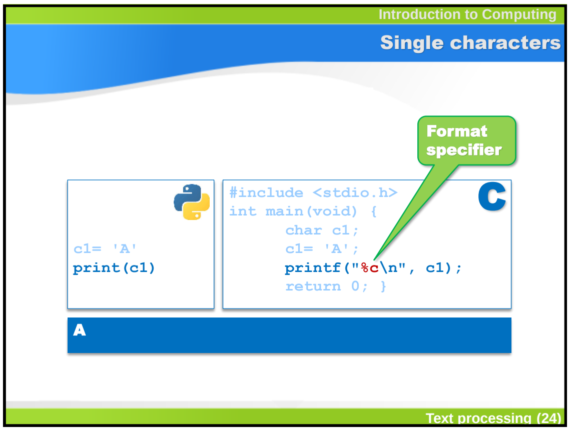
Assignment of a constant value

```
c1= 'A'
print(c1)
```

```
#include <stdio.h
int main(void)
      char c1;
      c1= 'A';
      printf("%c\n", c1);
      return 0; }
```

```
A
```

Text processing (23)



Introduction to Computing

Single characters

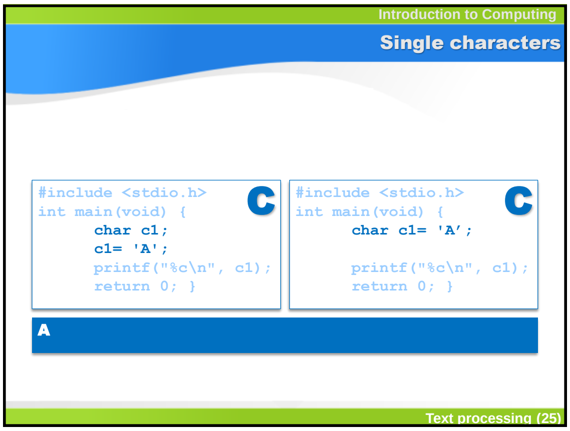
Format specifier

```
c1= 'A'
print(c1)
```

```
#include <stdio.h>
int main(void) {
      char c1;
      c1= 'A';
      printf("%c\n", c1);
      return 0; }
```

```
A
```

Text processing (24)



Introduction to Computing

Single characters

```
#include <stdio.h>
int main(void) {
        char c1;
        c1= 'A';
        printf("%c\n", c1);
        return 0; }
```

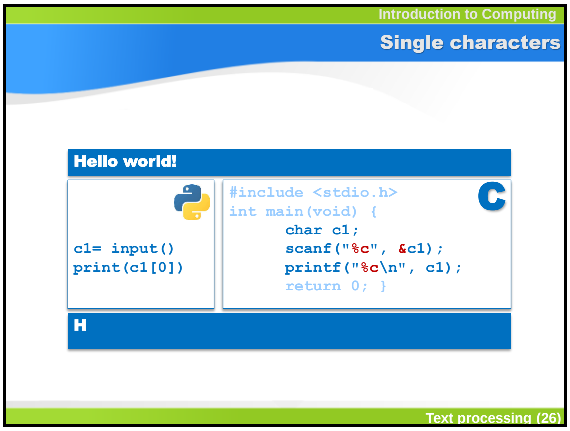
C

```
#include <stdio.h>
int main(void) {
        char c1= 'A';

        printf("%c\n", c1);
        return 0; }
```

C

```
A
```

Text processing (25)



Introduction to Computing

Single characters

Hello world!

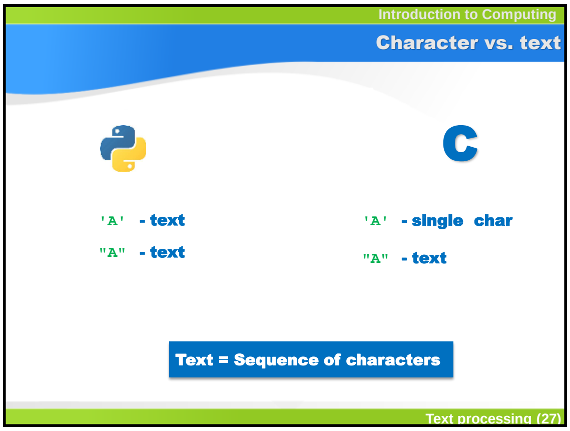
```
c1= input()
print(c1[0])
```

```
#include <stdio.h>
int main(void) {
        char c1;
        scanf("%c", &c1);
        printf("%c\n", c1);
        return 0; }
```

C

```
H
```

Text processing (26)



Introduction to Computing

Character vs. text

C

`'A'` - text

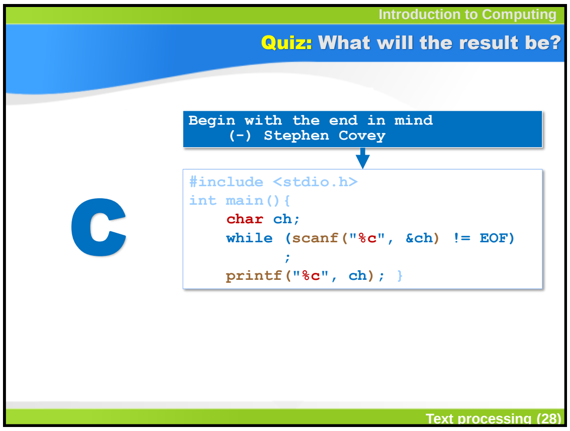
`"A"` - text

`'A'` - single char

`"A"` - text

Text = Sequence of characters

Text processing (27)



Introduction to Computing

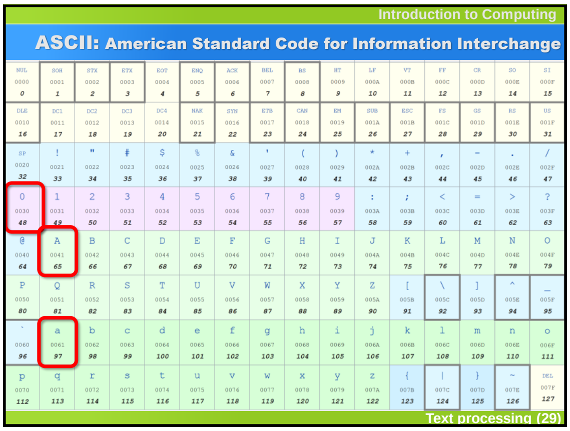
Quiz: What will the result be?

```
Begin with the end in mind
     (-) Stephen Covey
```

C

```
#include <stdio.h>
int main(){
     char ch;
     while (scanf("%c", &ch) != EOF)
          ;
     printf("%c", ch); }
```

Text processing (28)

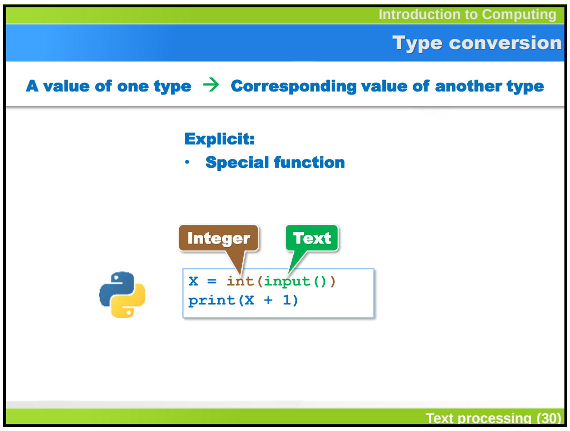


Introduction to Computing

ASCII: American Standard Code for Information Interchange

NUL 0000 0	SOH 0001 1	STX 0002 2	ETX 0003 3	EOT 0004 4	ENQ 0005 5	ACK 0006 6	BEL 0007 7	BS 0008 8	HT 0009 9	LF 000A 10	VT 000B 11	FF 000C 12	CR 000D 13	SO 000E 14	SI 000F 15
DLE 0010 16	DC1 0011 17	DC2 0012 18	DC3 0013 19	DC4 0014 20	NAK 0015 21	SYN 0016 22	ETB 0017 23	CAN 0018 24	EM 0019 25	SUB 001A 26	ESC 001B 27	FS 001C 28	GS 001D 29	RS 001E 30	US 001F 31
SP 0020 32	! 0021 33	" 0022 34	# 0023 35	$ 0024 36	% 0025 37	& 0026 38	' 0027 39	(0028 40	) 0029 41	* 002A 42	+ 002B 43	, 002C 44	- 002D 45	. 002E 46	/ 002F 47
0 0030 48	1 0031 49	2 0032 50	3 0033 51	4 0034 52	5 0035 53	6 0036 54	7 0037 55	8 0038 56	9 0039 57	: 003A 58	; 003B 59	< 003C 60	= 003D 61	> 003E 62	? 003F 63
@ 0040 64	A 0041 65	B 0042 66	C 0043 67	D 0044 68	E 0045 69	F 0046 70	G 0047 71	H 0048 72	I 0049 73	J 004A 74	K 004B 75	L 004C 76	M 004D 77	N 004E 78	O 004F 79
P 0050 80	Q 0051 81	R 0052 82	S 0053 83	T 0054 84	U 0055 85	V 0056 86	W 0057 87	X 0058 88	Y 0059 89	Z 005A 90	[005B 91	\ 005C 92	] 005D 93	^ 005E 94	_ 005F 95
` 0060 96	a 0061 97	b 0062 98	c 0063 99	d 0064 100	e 0065 101	f 0066 102	g 0067 103	h 0068 104	i 0069 105	j 006A 106	k 006B 107	l 006C 108	m 006D 109	n 006E 110	o 006F 111
p 0070 112	q 0071 113	r 0072 114	s 0073 115	t 0074 116	u 0075 117	v 0076 118	w 0077 119	x 0078 120	y 0079 121	z 007A 122	{ 007B 123	| 007C 124	} 007D 125	~ 007E 126	DEL 007F 127

Text processing (29)



Introduction to Computing

Type conversion

A value of one type → Corresponding value of another type

Explicit:

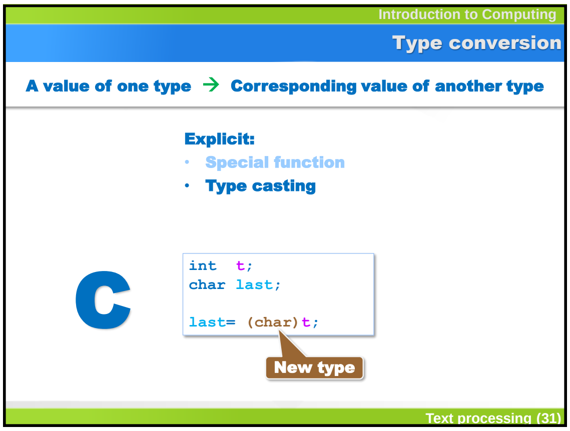
- **Special function**

Integer

Text

```
X = int(input())
print(X + 1)
```

Text processing (30)



Introduction to Computing

Type conversion

A value of one type → Corresponding value of another type

Explicit:

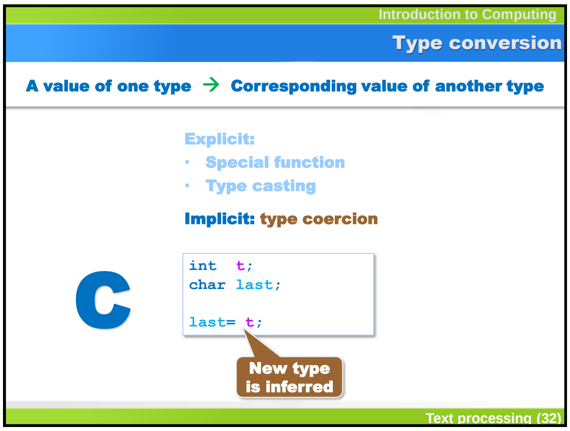
- Special function
- **Type casting**

C

```
int  t;
char last;

last= (char)t;
```

New type

Text processing (31)



Introduction to Computing

Type conversion

A value of one type → Corresponding value of another type

Explicit:

- Special function
- Type casting

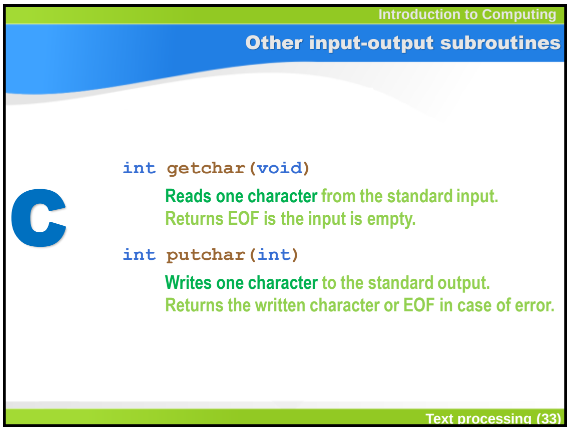
Implicit: type coercion

C

```
int  t;
char last;

last= t;
```

New type is inferred

Text processing (32)



Introduction to Computing

Other input-output subroutines

C

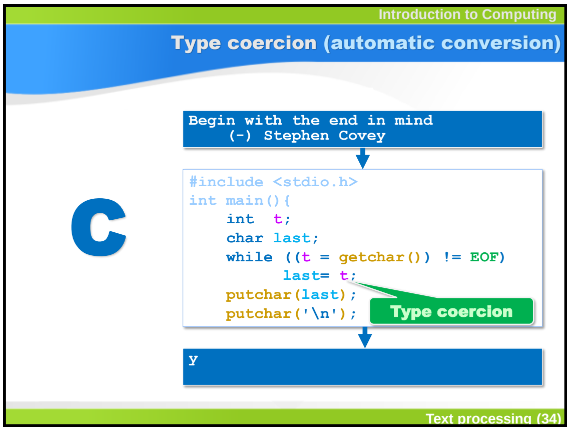
```
int getchar(void)
```

Reads one character from the standard input.
Returns EOF is the input is empty.

```
int putchar(int)
```

Writes one character to the standard output.
Returns the written character or EOF in case of error.

Text processing (33)



Introduction to Computing

Type coercion (automatic conversion)

C

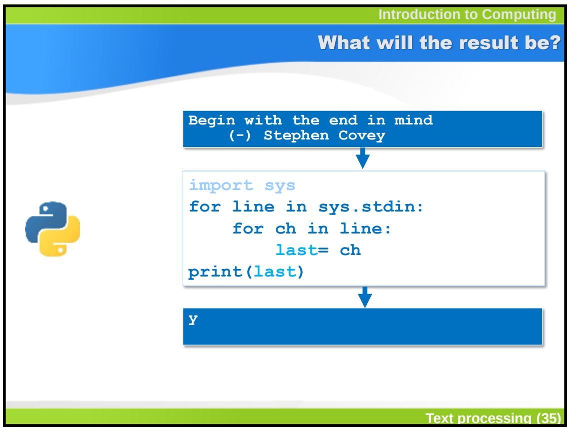
```
Begin with the end in mind
    (-) Stephen Covey
```

```
#include <stdio.h>
int main(){
    int  t;
    char last;
    while ((t = getchar()) != EOF)
         last= t;
    putchar(last);
    putchar('\n');
```

Type coercion

```
y
```

Text processing (34)



Introduction to Computing

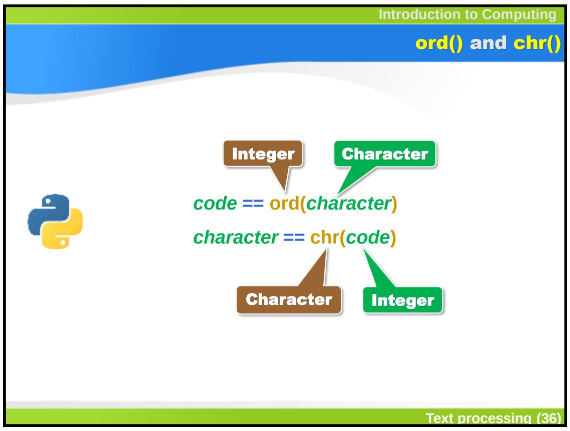
What will the result be?

```
Begin with the end in mind
    (-) Stephen Covey
```

```
import sys
for line in sys.stdin:
    for ch in line:
        last= ch
print(last)
```

```
y
```

Text processing (35)



Introduction to Computing

ord() and chr()

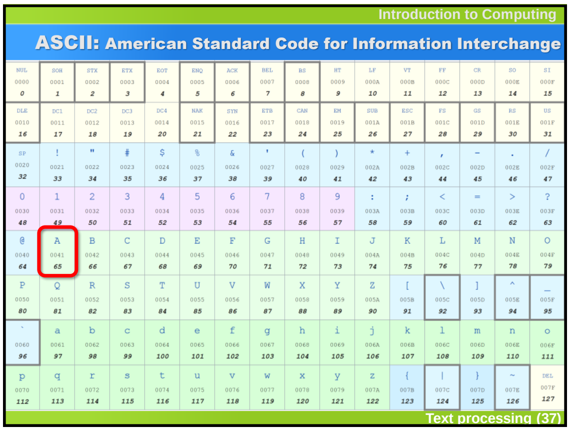
Integer Character

code == ord(*character*)

character == chr(*code*)

Character Integer

Text processing (36)

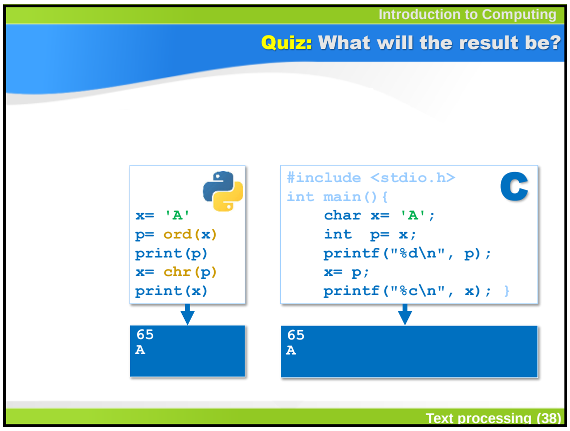


Introduction to Computing

ASCII: American Standard Code for Information Interchange

NUL 0000 0	SOH 0001 1	STX 0002 2	ETX 0003 3	EOT 0004 4	ENQ 0005 5	ACK 0006 6	BEL 0007 7	BS 0008 8	HT 0009 9	LF 000A 10	VT 000B 11	FF 000C 12	CR 000D 13	SO 000E 14	SI 000F 15
DLE 0010 16	DC1 0011 17	DC2 0012 18	DC3 0013 19	DC4 0014 20	NAK 0015 21	SYN 0016 22	ETB 0017 23	CAN 0018 24	EM 0019 25	SUB 001A 26	ESC 001B 27	FS 001C 28	GS 001D 29	RS 001E 30	US 001F 31
SP 0020 32	! 0021 33	" 0022 34	# 0023 35	$ 0024 36	% 0025 37	& 0026 38	' 0027 39	(0028 40	) 0029 41	* 002A 42	+ 002B 43	, 002C 44	- 002D 45	. 002E 46	/ 002F 47
0 0030 48	1 0031 49	2 0032 50	3 0033 51	4 0034 52	5 0035 53	6 0036 54	7 0037 55	8 0038 56	9 0039 57	: 003A 58	; 003B 59	< 003C 60	= 003D 61	> 003E 62	? 003F 63
@ 0040 64	A 0041 65	B 0042 66	C 0043 67	D 0044 68	E 0045 69	F 0046 70	G 0047 71	H 0048 72	I 0049 73	J 004A 74	K 004B 75	L 004C 76	M 004D 77	N 004E 78	O 004F 79
P 0050 80	Q 0051 81	R 0052 82	S 0053 83	T 0054 84	U 0055 85	V 0056 86	W 0057 87	X 0058 88	Y 0059 89	Z 005A 90	[005B 91	\ 005C 92	] 005D 93	^ 005E 94	_ 005F 95
` 0060 96	a 0061 97	b 0062 98	c 0063 99	d 0064 100	e 0065 101	f 0066 102	g 0067 103	h 0068 104	i 0069 105	j 006A 106	k 006B 107	l 006C 108	m 006D 109	n 006E 110	o 006F 111
p 0070 112	q 0071 113	r 0072 114	s 0073 115	t 0074 116	u 0075 117	v 0076 118	w 0077 119	x 0078 120	y 0079 121	z 007A 122	{ 007B 123	| 007C 124	} 007D 125	~ 007E 126	DEL 007F 127

Text processing (37)



Introduction to Computing

Quiz: What will the result be?

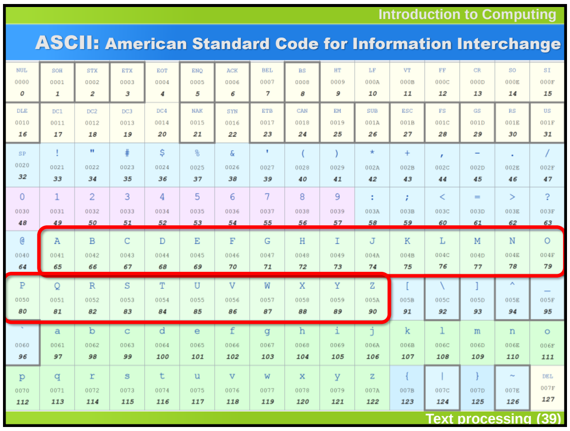
```
x= 'A'
p= ord(x)
print(p)
x= chr(p)
print(x)
```

```
65
A
```

```
#include <stdio.h>
int main(){
    char x= 'A';
    int  p= x;
    printf("%d\n", p);
    x= p;
    printf("%c\n", x); }
```

```
65
A
```

Text processing (38)

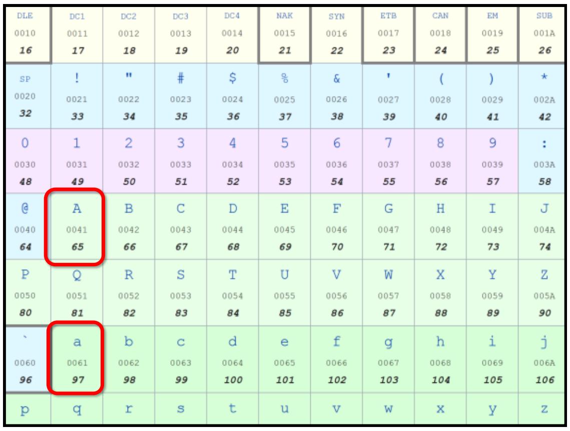


Introduction to Computing

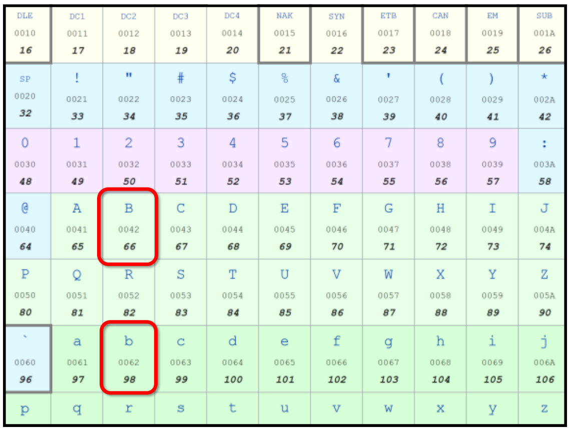
ASCII: American Standard Code for Information Interchange

NUL 0000 0	SOH 0001 1	STX 0002 2	ETX 0003 3	EOT 0004 4	ENQ 0005 5	ACK 0006 6	BEL 0007 7	BS 0008 8	HT 0009 9	LF 000A 10	VT 000B 11	FF 000C 12	CR 000D 13	SO 000E 14	SI 000F 15
DLE 0010 16	DC1 0011 17	DC2 0012 18	DC3 0013 19	DC4 0014 20	NAK 0015 21	SYN 0016 22	ETB 0017 23	CAN 0018 24	EM 0019 25	SUB 001A 26	ESC 001B 27	FS 001C 28	GS 001D 29	RS 001E 30	US 001F 31
SP 0020 32	! 0021 33	" 0022 34	# 0023 35	$ 0024 36	% 0025 37	& 0026 38	' 0027 39	(0028 40	) 0029 41	* 002A 42	+ 002B 43	, 002C 44	- 002D 45	. 002E 46	/ 002F 47
0 0030 48	1 0031 49	2 0032 50	3 0033 51	4 0034 52	5 0035 53	6 0036 54	7 0037 55	8 0038 56	9 0039 57	: 003A 58	; 003B 59	< 003C 60	= 003D 61	> 003E 62	? 003F 63
@ 0040 64	A 0041 65	B 0042 66	C 0043 67	D 0044 68	E 0045 69	F 0046 70	G 0047 71	H 0048 72	I 0049 73	J 004A 74	K 004B 75	L 004C 76	M 004D 77	N 004E 78	O 004F 79
P 0050 80	Q 0051 81	R 0052 82	S 0053 83	T 0054 84	U 0055 85	V 0056 86	W 0057 87	X 0058 88	Y 0059 89	Z 005A 90	[005B 91	\ 005C 92	] 005D 93	^ 005E 94	_ 005F 95
` 0060 96	a 0061 97	b 0062 98	c 0063 99	d 0064 100	e 0065 101	f 0066 102	g 0067 103	h 0068 104	i 0069 105	j 006A 106	k 006B 107	l 006C 108	m 006D 109	n 006E 110	o 006F 111
p 0070 112	q 0071 113	r 0072 114	s 0073 115	t 0074 116	u 0075 117	v 0076 118	w 0077 119	x 0078 120	y 0079 121	z 007A 122	{ 007B 123	| 007C 124	} 007D 125	~ 007E 126	DEL 007F 127

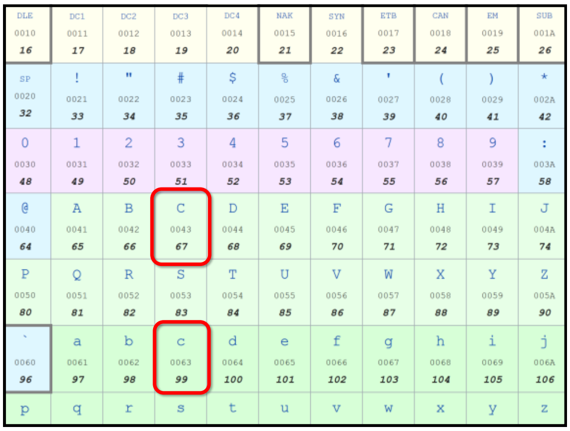
Text processing (39)



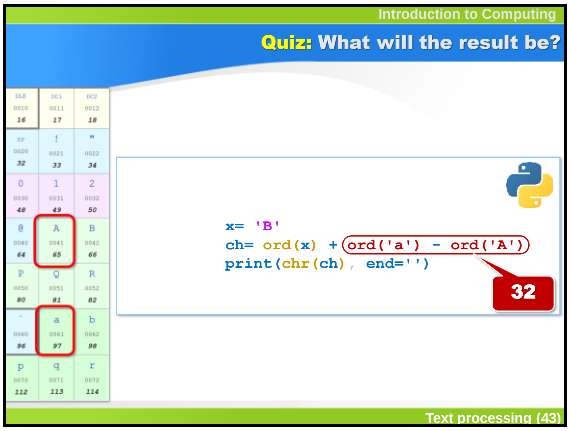
DLE 0010 16	DC1 0011 17	DC2 0012 18	DC3 0013 19	DC4 0014 20	NAK 0015 21	SYN 0016 22	ETB 0017 23	CAN 0018 24	EM 0019 25	SUB 001A 26
SP 0020 32	! 0021 33	" 0022 34	# 0023 35	$ 0024 36	% 0025 37	& 0026 38	' 0027 39	(0028 40	) 0029 41	* 002A 42
0 0030 48	1 0031 49	2 0032 50	3 0033 51	4 0034 52	5 0035 53	6 0036 54	7 0037 55	8 0038 56	9 0039 57	: 003A 58
@ 0040 64	A 0041 65	B 0042 66	C 0043 67	D 0044 68	E 0045 69	F 0046 70	G 0047 71	H 0048 72	I 0049 73	J 004A 74
P 0050 80	Q 0051 81	R 0052 82	S 0053 83	T 0054 84	U 0055 85	V 0056 86	W 0057 87	X 0058 88	Y 0059 89	Z 005A 90
` 0060 96	a 0061 97	b 0062 98	c 0063 99	d 0064 100	e 0065 101	f 0066 102	g 0067 103	h 0068 104	i 0069 105	j 006A 106
p	q	r	s	t	u	v	w	x	y	z



DLE 0010 16	DC1 0011 17	DC2 0012 18	DC3 0013 19	DC4 0014 20	NAK 0015 21	SYN 0016 22	ETB 0017 23	CAN 0018 24	EM 0019 25	SUB 001A 26
SP 0020 32	! 0021 33	" 0022 34	# 0023 35	$ 0024 36	% 0025 37	& 0026 38	' 0027 39	(0028 40	) 0029 41	* 002A 42
0 0030 48	1 0031 49	2 0032 50	3 0033 51	4 0034 52	5 0035 53	6 0036 54	7 0037 55	8 0038 56	9 0039 57	: 003A 58
@ 0040 64	A 0041 65	B 0042 66	C 0043 67	D 0044 68	E 0045 69	F 0046 70	G 0047 71	H 0048 72	I 0049 73	J 004A 74
P 0050 80	Q 0051 81	R 0052 82	S 0053 83	T 0054 84	U 0055 85	V 0056 86	W 0057 87	X 0058 88	Y 0059 89	Z 005A 90
` 0060 96	a 0061 97	b 0062 98	c 0063 99	d 0064 100	e 0065 101	f 0066 102	g 0067 103	h 0068 104	i 0069 105	j 006A 106
p	q	r	s	t	u	v	w	x	y	z



DLE 0010 16	DC1 0011 17	DC2 0012 18	DC3 0013 19	DC4 0014 20	NAK 0015 21	SYN 0016 22	ETB 0017 23	CAN 0018 24	EM 0019 25	SUB 001A 26
SP 0020 32	! 0021 33	" 0022 34	# 0023 35	$ 0024 36	% 0025 37	& 0026 38	' 0027 39	(0028 40	) 0029 41	* 002A 42
0 0030 48	1 0031 49	2 0032 50	3 0033 51	4 0034 52	5 0035 53	6 0036 54	7 0037 55	8 0038 56	9 0039 57	: 003A 58
@ 0040 64	A 0041 65	B 0042 66	C 0043 67	D 0044 68	E 0045 69	F 0046 70	G 0047 71	H 0048 72	I 0049 73	J 004A 74
P 0050 80	Q 0051 81	R 0052 82	S 0053 83	T 0054 84	U 0055 85	V 0056 86	W 0057 87	X 0058 88	Y 0059 89	Z 005A 90
` 0060 96	a 0061 97	b 0062 98	c 0063 99	d 0064 100	e 0065 101	f 0066 102	g 0067 103	h 0068 104	i 0069 105	j 006A 106
p	q	r	s	t	u	v	w	x	y	z

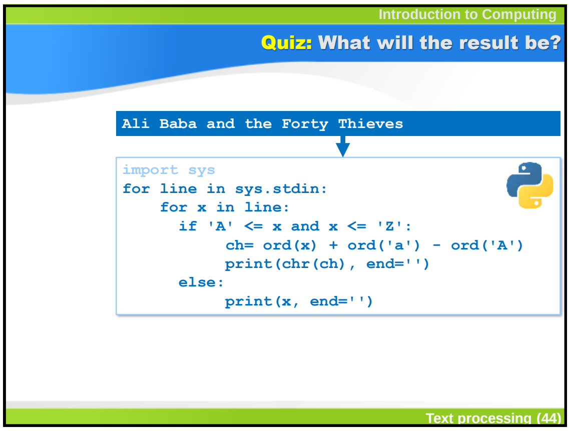


Quiz: What will the result be?

DLE 0010 16	DC1 0011 17	DC2 0012 18
SP 0020 32	! 0021 33	" 0022 34
0 0030 48	1 0031 49	2 0032 50
@ 0040 64	A 0041 65	B 0042 66
P 0050 80	Q 0051 81	R 0052 82
` 0060 96	a 0061 97	b 0062 98
p 0070 112	q 0071 113	r 0072 114

```
x= 'B'
ch= ord(x) + ord('a') - ord('A')
print(chr(ch), end='')
```

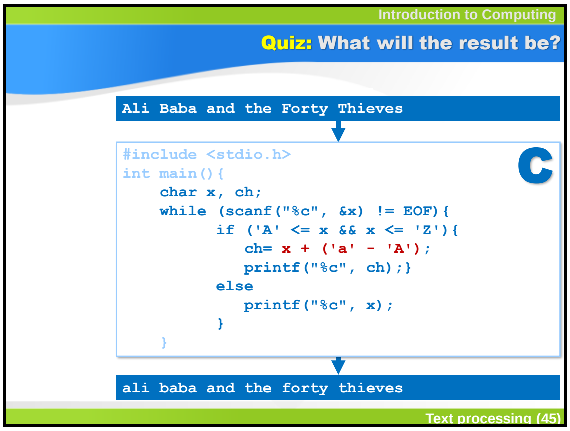
32



Quiz: What will the result be?

```
Ali Baba and the Forty Thieves
```

```
import sys
for line in sys.stdin:
    for x in line:
      if 'A' <= x and x <= 'Z':
          ch= ord(x) + ord('a') - ord('A')
          print(chr(ch), end='')
      else:
          print(x, end='')
```



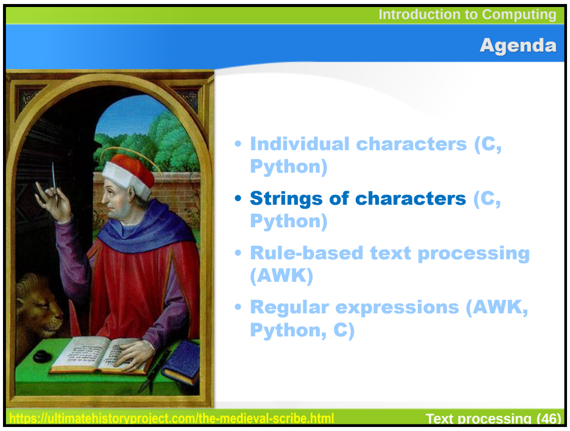
Quiz: What will the result be?

```
Ali Baba and the Forty Thieves
```

C

```
#include <stdio.h>
int main(){
    char x, ch;
    while (scanf("%c", &x) != EOF){
          if ('A' <= x && x <= 'Z'){
             ch= x + ('a' - 'A');
             printf("%c", ch);}
          else
             printf("%c", x);
          }
    }
```

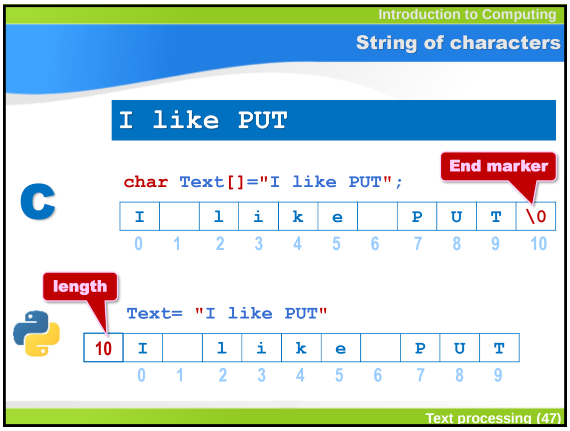
```
ali baba and the forty thieves
```



Agenda

- Individual characters (C, Python)
- **Strings of characters** (C, Python)
- Rule-based text processing (AWK)
- Regular expressions (AWK, Python, C)

https://ultimatehistoryproject.com/the-medieval-scribe.html



String of characters

```
I like PUT
```

C

```
char Text[]="I like PUT";
```

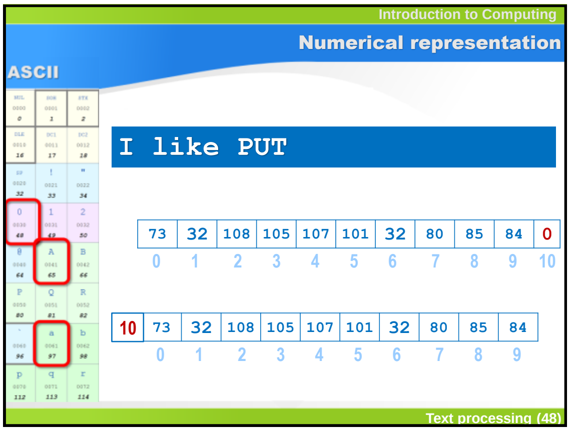
End marker

I		l	i	k	e		P	U	T	\0
0	1	2	3	4	5	6	7	8	9	10

length

```
Text= "I like PUT"
```

10	I		l	i	k	e		P	U	T
	0	1	2	3	4	5	6	7	8	9



Numerical representation

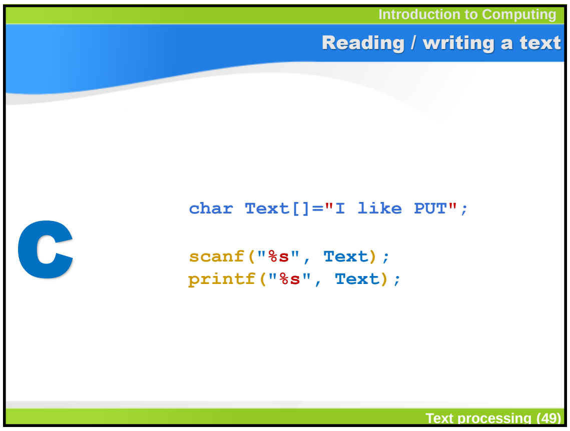
ASCII

NUL 0000 0	SOH 0001 1	STX 0002 2
DLE 0010 16	DC1 0011 17	DC2 0012 18
SP 0020 32	! 0021 33	" 0022 34
0 0030 48	1 0031 49	2 0032 50
@ 0040 64	A 0041 65	B 0042 66
P 0050 80	Q 0051 81	R 0052 82
` 0060 96	a 0061 97	b 0062 98
p 0070 112	q 0071 113	r 0072 114

```
I like PUT
```

73	32	108	105	107	101	32	80	85	84	0
0	1	2	3	4	5	6	7	8	9	10

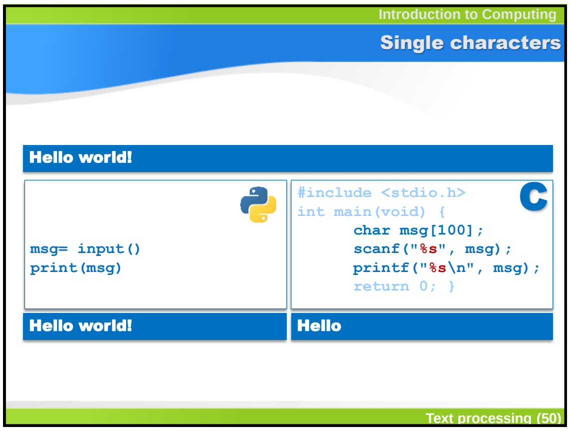
10	73	32	108	105	107	101	32	80	85	84
	0	1	2	3	4	5	6	7	8	9



Reading / writing a text

C

```
char Text[]="I like PUT";

scanf("%s", Text);
printf("%s", Text);
```



Single characters

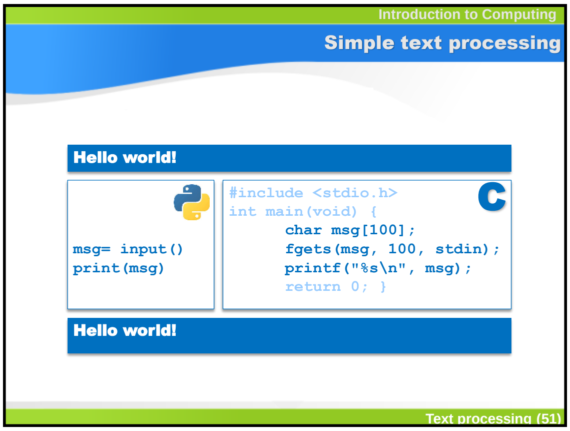
Hello world!

```
msg= input()
print(msg)
```

```
#include <stdio.h>
int main(void) {
      char msg[100];
      scanf("%s", msg);
      printf("%s\n", msg);
      return 0; }
```

Hello world!

Hello



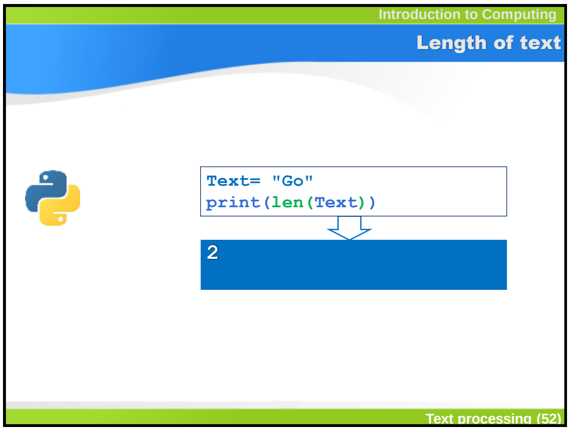
Simple text processing

Hello world!

```
msg= input()
print(msg)
```

```
#include <stdio.h>
int main(void) {
      char msg[100];
      fgets(msg, 100, stdin);
      printf("%s\n", msg);
      return 0; }
```

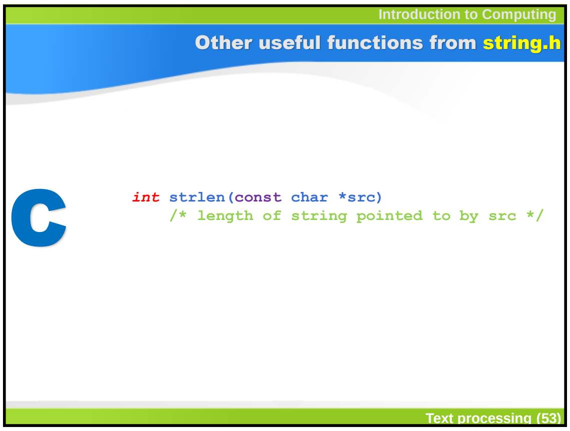
Hello world!



Length of text

```
Text= "Go"
print(len(Text))
```

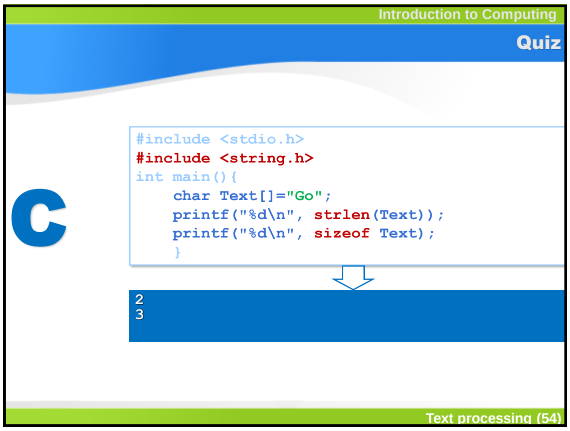
```
2
```



Other useful functions from string.h

C

```
int strlen(const char *src)
    /* length of string pointed to by src */
```



Quiz

C

```
#include <stdio.h>
#include <string.h>
int main(){
    char Text[]="Go";
    printf("%d\n", strlen(Text));
    printf("%d\n", sizeof Text);
    }
```

```
2
3
```

What will the result be?

```
Warsaw
```

```
Expected= "Warsaw"
Provided= input()
if Expected == Provided:
    print("OK")
else:
    print("Wrong!")
```

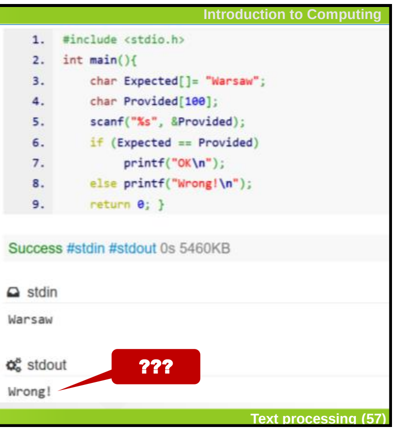
```
OK
```

What will the result be?

C

```
Warsaw
```

```
#include <stdio.h>
int main(){
    char Expected[]= "Warsaw";
    char Provided[100];
    scanf("%s", &Provided);
    if (Expected == Provided)
        printf("OK\n");
    else printf("Wrong!\n");}
```



```
#include <stdio.h>
int main(){
    char Expected[]= "Warsaw";
    char Provided[100];
    scanf("%s", &Provided);
    if (Expected == Provided)
        printf("OK\n");
    else printf("Wrong!\n");
    return 0; }
```

Success #stdin #stdout 0s 5460KB

stdin

```
Warsaw
```

stdout

```
Wrong!
```

What will the result be?

C

```
Warsaw
```

```
#include <string.h>
#include <stdio.h>
int main(){
    char Expected[]= "Warsaw";
    char Provided[100];
    scanf("%s", &Provided);
    if (!strcmp(Expected, Provided))
        printf("OK\n");
    else printf("Wrong!\n");
    return 0; }
```

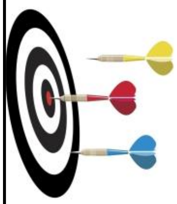
```
OK
```

Agenda

- Individual characters (C, Python)
- Strings of characters (C, Python)
- **Rule-based text processing (AWK)**
- Regular expressions (AWK, Python, C)

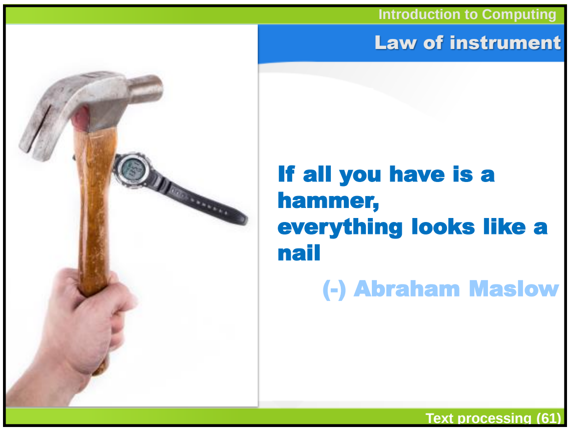
https://ultimatehistoryproject.com/the-medieval-scribe.html

Aim of the lecture



To present:

- Another programming paradigm: **rule-based programming**
- **Rudiments of AWK**



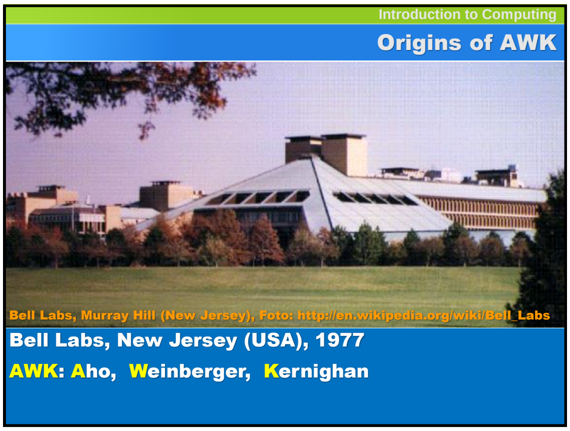
Introduction to Computing

Law of instrument

If all you have is a hammer,
everything looks like a nail

(-) Abraham Maslow

Text processing (61)



Introduction to Computing

Origins of AWK

Bell Labs, Murray Hill (New Jersey), Foto: http://en.wikipedia.org/wiki/Bell_Labs

Bell Labs, New Jersey (USA), 1977

AWK: Aho, Weinberger, Kernighan

Introduction to Computing

Authors of AWK

Alfred Aho

Peter Weinberger

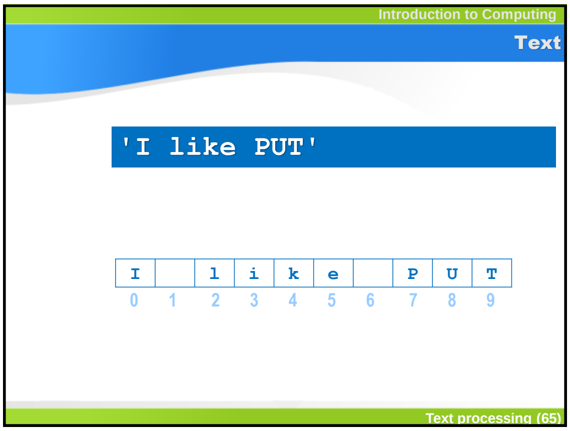
Brian Kernighan

http://www.underforty.us/geeks.html

Text processing (63)

Introduction to Computing

Fundamental question



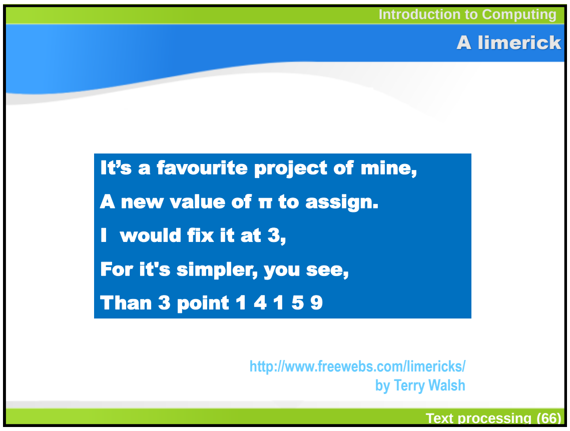
Introduction to Computing

Text

'I like PUT'

I		l	i	k	e		P	U	T
0	1	2	3	4	5	6	7	8	9

Text processing (65)



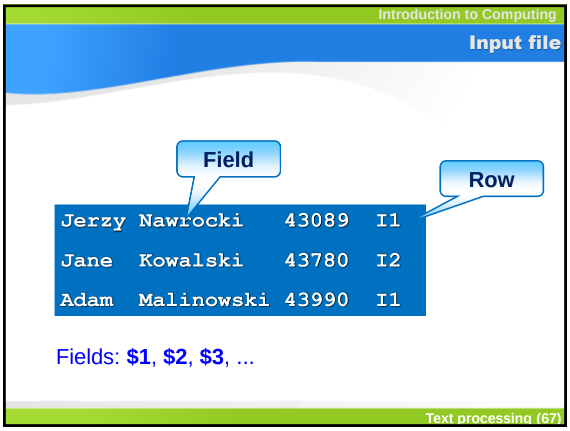
Introduction to Computing

A limerick

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

http://www.freewebs.com/limericks/
by Terry Walsh

Text processing (66)



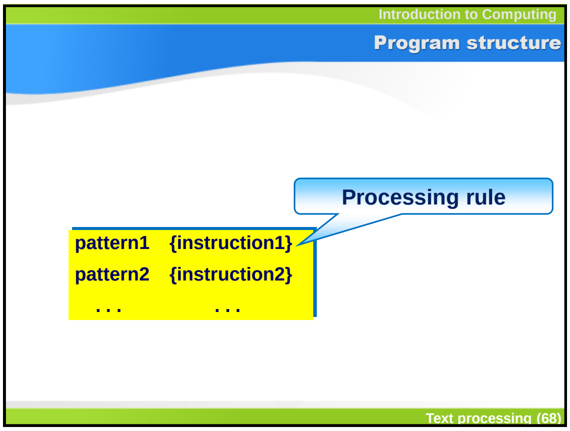
Input file

Field

Row

```
Jerzy Nawrocki   43089  I1
Jane  Kowalski   43780  I2
Adam  Malinowski 43990  I1
```

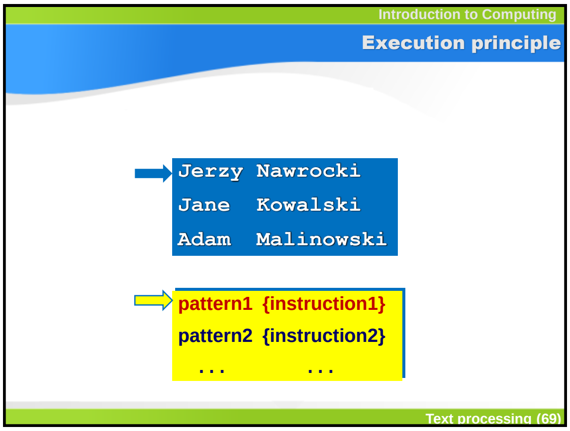
Fields: **$1**, **$2**, **$3**, ...



Program structure

Processing rule

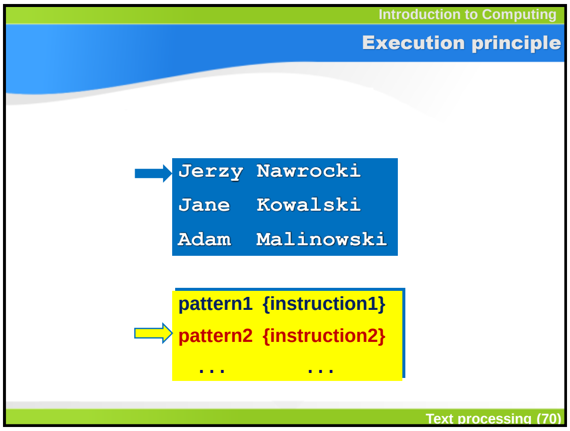
```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```



Execution principle

```
Jerzy Nawrocki
Jane  Kowalski
Adam  Malinowski
```

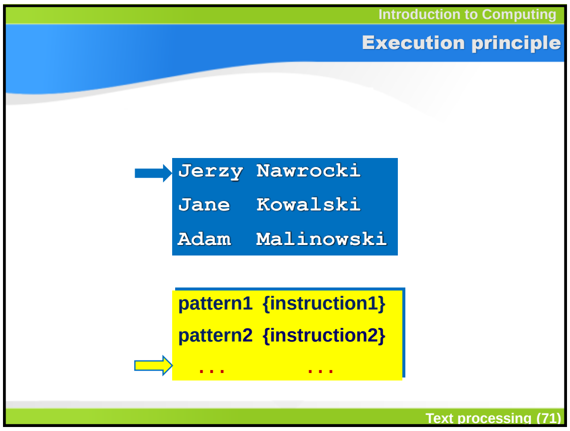
```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```



Execution principle

```
Jerzy Nawrocki
Jane  Kowalski
Adam  Malinowski
```

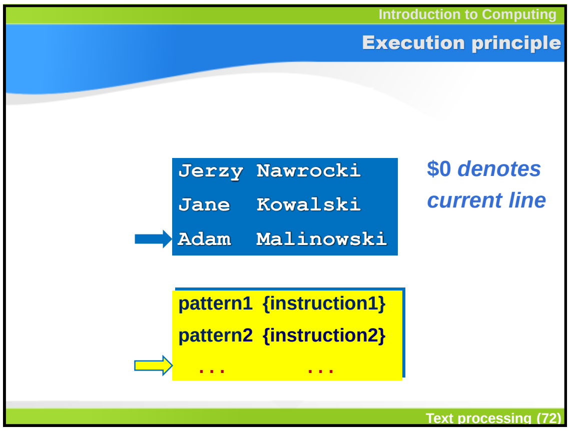
```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```



Execution principle

```
Jerzy Nawrocki
Jane  Kowalski
Adam  Malinowski
```

```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```

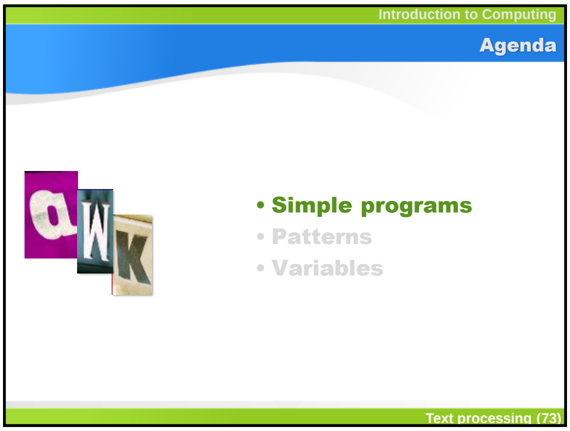


Execution principle

```
Jerzy Nawrocki
Jane  Kowalski
Adam  Malinowski
```

$0 denotes current line

```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```

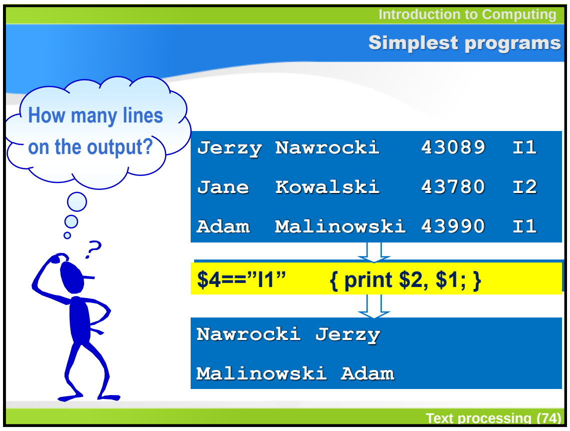


Introduction to Computing

Agenda

- **Simple programs**
- Patterns
- Variables

Text processing (73)



Introduction to Computing

Simplest programs

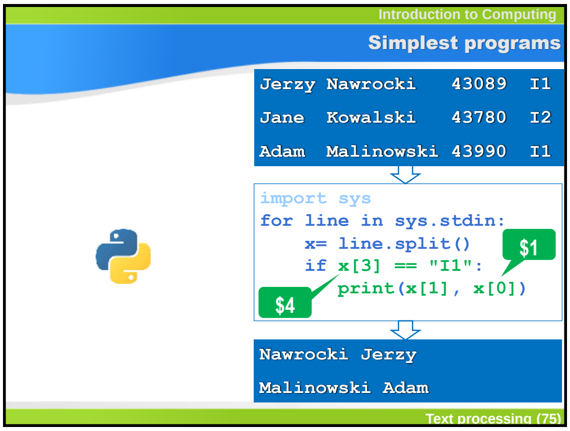
How many lines on the output?

```
Jerzy Nawrocki    43089  I1
Jane  Kowalski    43780  I2
Adam  Malinowski  43990  I1
```

$4==”I1”   { print $2, $1; }

```
Nawrocki Jerzy
Malinowski Adam
```

Text processing (74)



Introduction to Computing

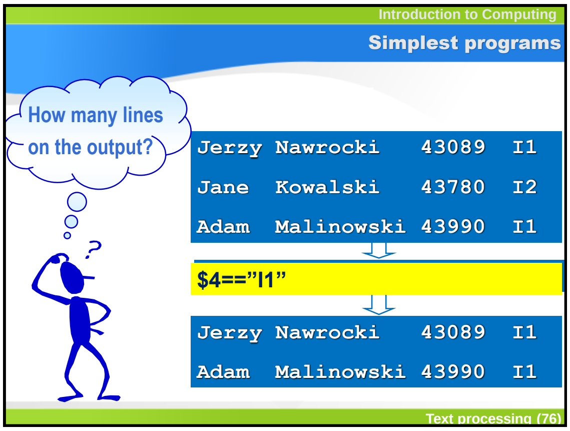
Simplest programs

```
Jerzy Nawrocki    43089  I1
Jane  Kowalski    43780  I2
Adam  Malinowski  43990  I1
```

```
import sys
for line in sys.stdin:
    x= line.split()
    if x[3] == "I1":
        print(x[1], x[0])
```

```
Nawrocki Jerzy
Malinowski Adam
```

Text processing (75)



Introduction to Computing

Simplest programs

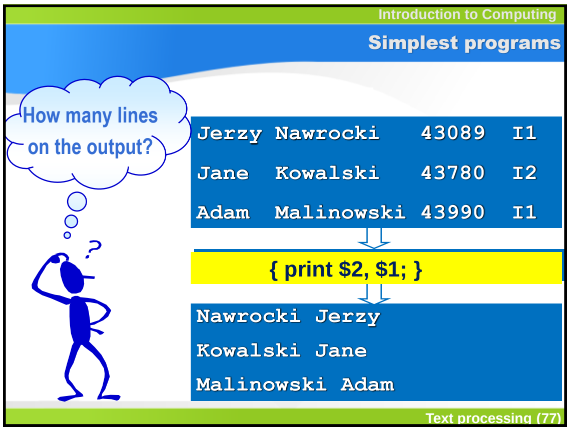
How many lines on the output?

```
Jerzy Nawrocki    43089  I1
Jane  Kowalski    43780  I2
Adam  Malinowski  43990  I1
```

$4==”I1”

```
Jerzy Nawrocki    43089  I1
Adam  Malinowski  43990  I1
```

Text processing (76)



Introduction to Computing

Simplest programs

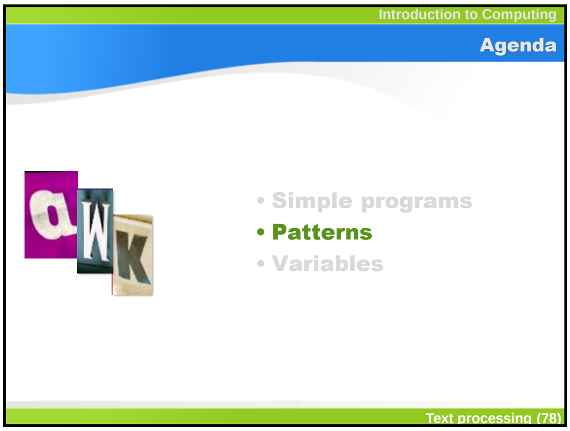
How many lines on the output?

```
Jerzy Nawrocki    43089  I1
Jane  Kowalski    43780  I2
Adam  Malinowski  43990  I1
```

{ print $2, $1; }

```
Nawrocki Jerzy
Kowalski Jane
Malinowski Adam
```

Text processing (77)

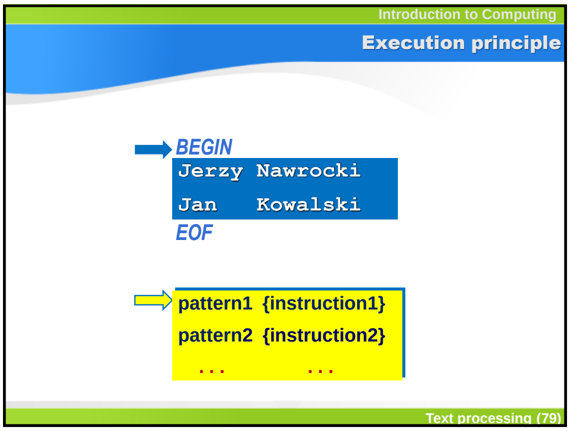


Introduction to Computing

Agenda

- Simple programs
- **Patterns**
- Variables

Text processing (78)



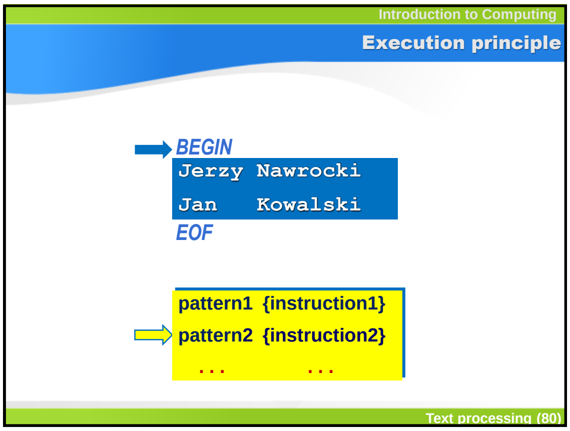
Introduction to Computing

Execution principle

```
BEGIN
Jerzy Nawrocki
Jan   Kowalski
EOF
```

```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```

Text processing (79)



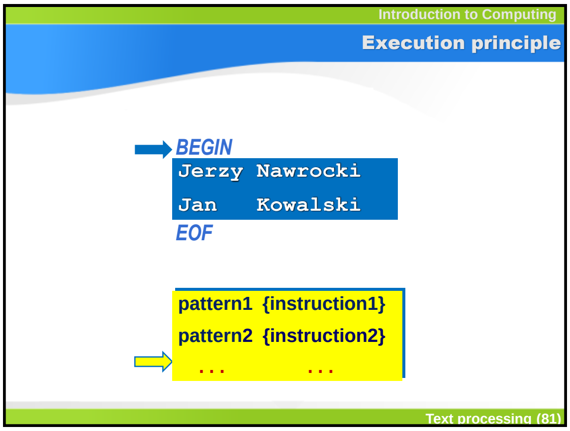
Introduction to Computing

Execution principle

```
BEGIN
Jerzy Nawrocki
Jan   Kowalski
EOF
```

```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```

Text processing (80)



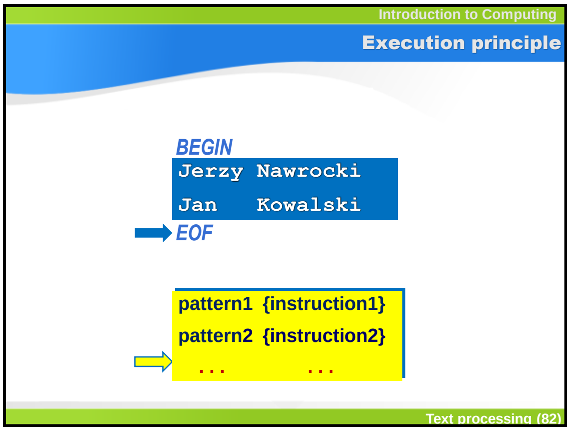
Introduction to Computing

Execution principle

```
BEGIN
Jerzy Nawrocki
Jan   Kowalski
EOF
```

```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```

Text processing (81)



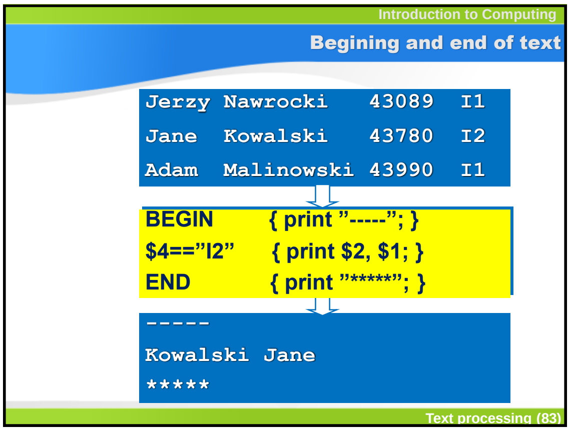
Introduction to Computing

Execution principle

```
BEGIN
Jerzy Nawrocki
Jan   Kowalski
EOF
```

```
pattern1  {instruction1}
pattern2  {instruction2}
  ...        ...
```

Text processing (82)



Introduction to Computing

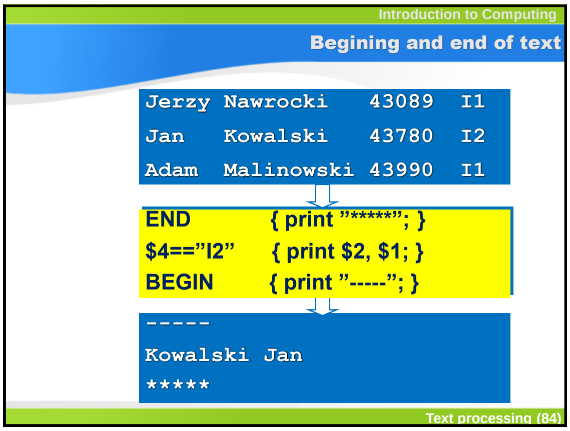
Begining and end of text

```
Jerzy Nawrocki  43089  I1
Jane  Kowalski  43780  I2
Adam  Malinowski 43990 I1
```

```
BEGIN       { print ”-----”; }
$4==”I2”    { print $2, $1; }
END         { print ”*****”; }
```

```
-----
Kowalski Jane
*****
```

Text processing (83)



Introduction to Computing

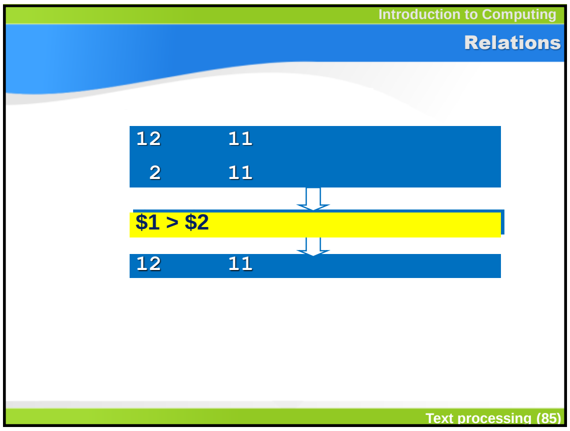
Begining and end of text

```
Jerzy Nawrocki  43089  I1
Jan   Kowalski  43780  I2
Adam  Malinowski 43990 I1
```

```
END         { print ”*****”; }
$4==”I2”    { print $2, $1; }
BEGIN       { print ”-----”; }
```

```
-----
Kowalski Jan
*****
```

Text processing (84)



Introduction to Computing

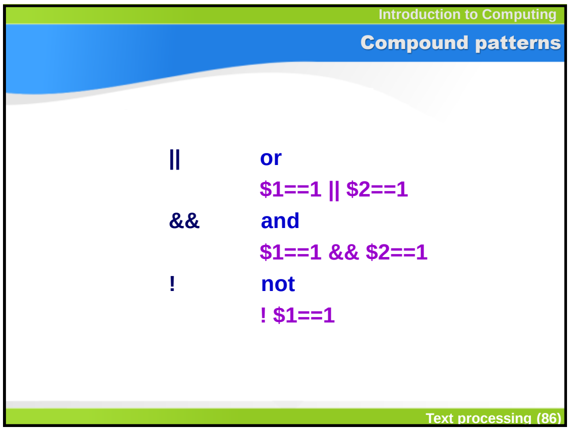
Relations

```
12      11
 2      11
```

$1 > $2

```
12      11
```

Text processing (85)

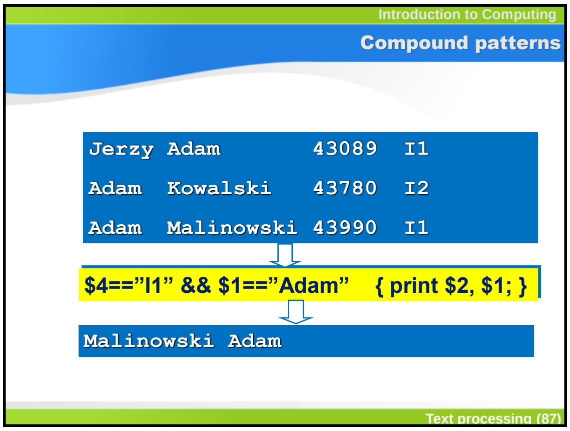


Introduction to Computing

Compound patterns

```
||      or
        $1==1 || $2==1
&&      and
        $1==1 && $2==1
!       not
        ! $1==1
```

Text processing (86)



Introduction to Computing

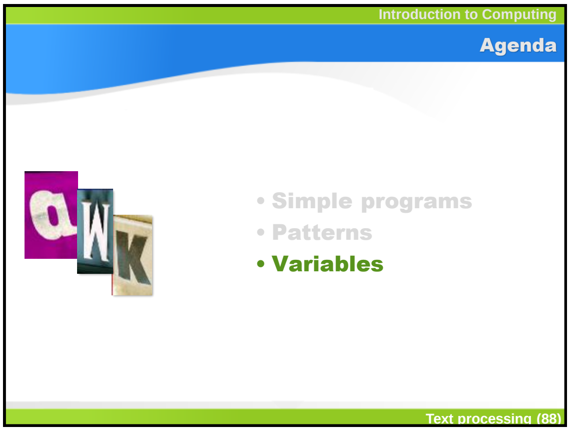
Compound patterns

```
Jerzy Adam       43089  I1
Adam  Kowalski   43780  I2
Adam  Malinowski 43990  I1
```

```
$4==”I1” && $1==”Adam”    { print $2, $1; }
```

```
Malinowski Adam
```

Text processing (87)

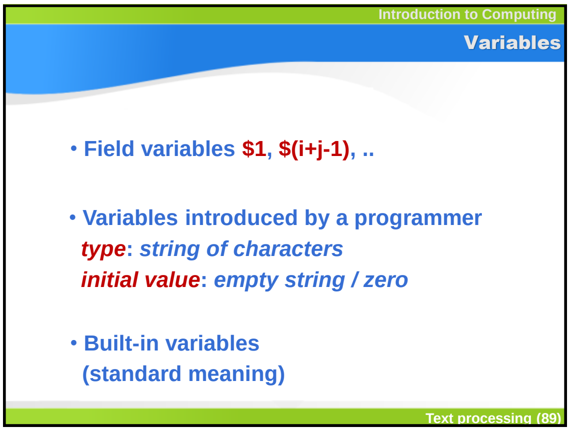


Introduction to Computing

Agenda

- Simple programs
- Patterns
- **Variables**

Text processing (88)

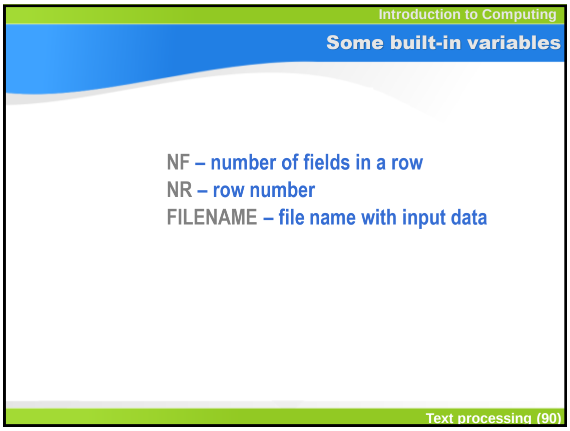


Introduction to Computing

Variables

- Field variables $1, $(i+j-1), ..
- Variables introduced by a programmer
  type: *string of characters*
  initial value: *empty string / zero*
- Built-in variables
  (standard meaning)

Text processing (89)

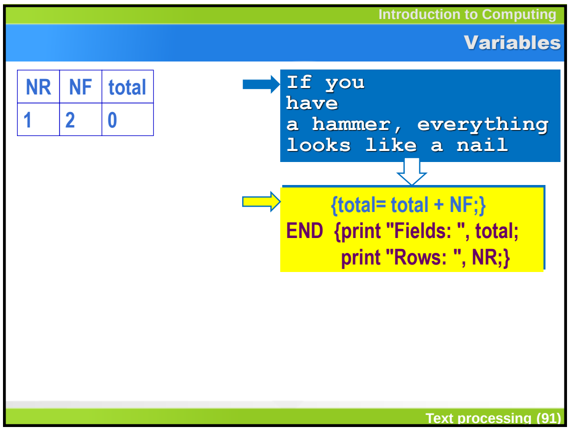


Introduction to Computing

Some built-in variables

NF – number of fields in a row
NR – row number
FILENAME – file name with input data

Text processing (90)



Introduction to Computing

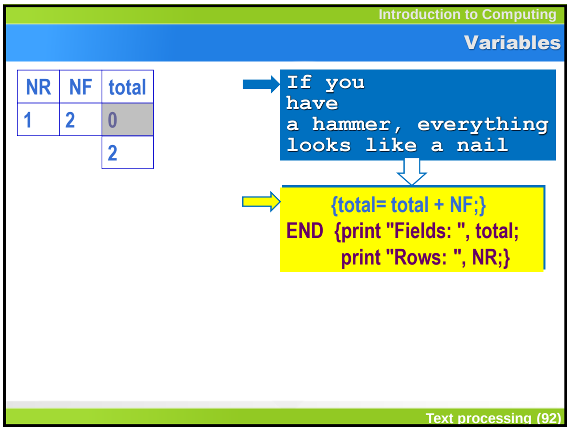
Variables

NR	NF	total
1	2	0

```
If you
have
a hammer, everything
looks like a nail
```

```
    {total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (91)



Introduction to Computing

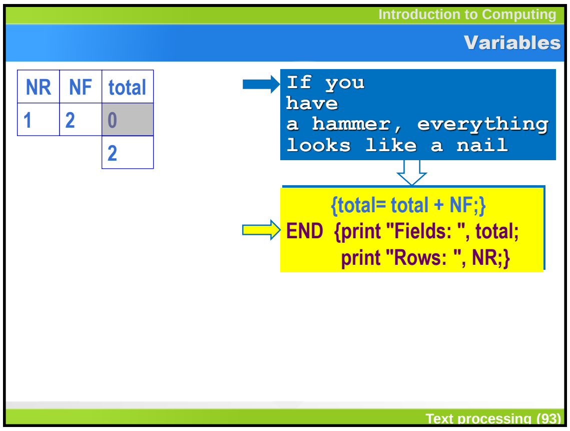
Variables

NR	NF	total
1	2	0
		2

```
If you
have
a hammer, everything
looks like a nail
```

```
    {total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (92)



Introduction to Computing

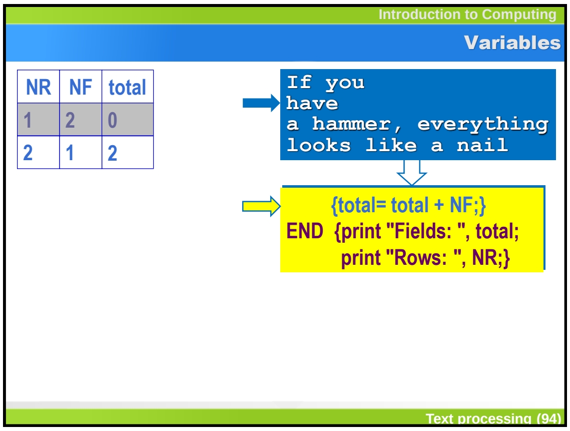
Variables

NR	NF	total
1	2	0
		2

```
If you
have
a hammer, everything
looks like a nail
```

```
    {total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (93)



Introduction to Computing

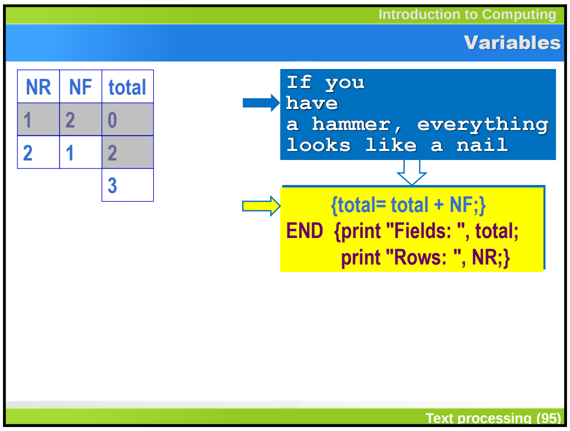
Variables

NR	NF	total
1	2	0
2	1	2

```
If you
have
a hammer, everything
looks like a nail
```

```
    {total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (94)



Introduction to Computing

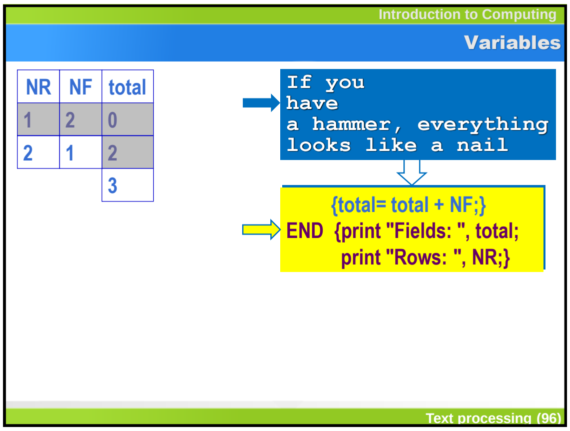
Variables

NR	NF	total
1	2	0
2	1	2
		3

```
If you
have
a hammer, everything
looks like a nail
```

```
    {total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (95)



Introduction to Computing

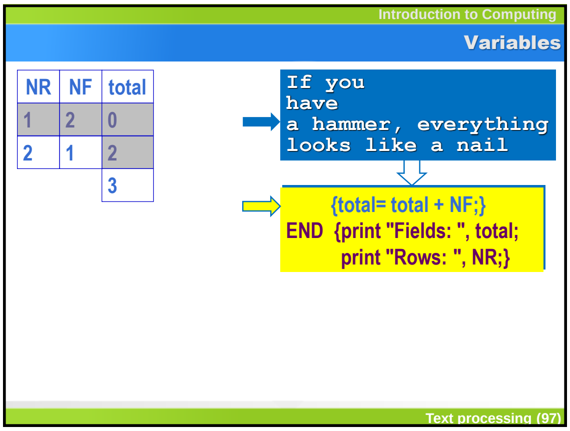
Variables

NR	NF	total
1	2	0
2	1	2
		3

```
If you
have
a hammer, everything
looks like a nail
```

```
    {total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (96)



Introduction to Computing

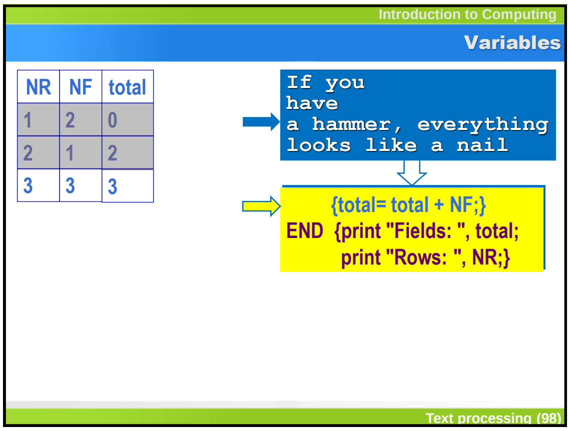
Variables

NR	NF	total
1	2	0
2	1	2
		3

```
If you
have
a hammer, everything
looks like a nail
```

```
{total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (97)



Introduction to Computing

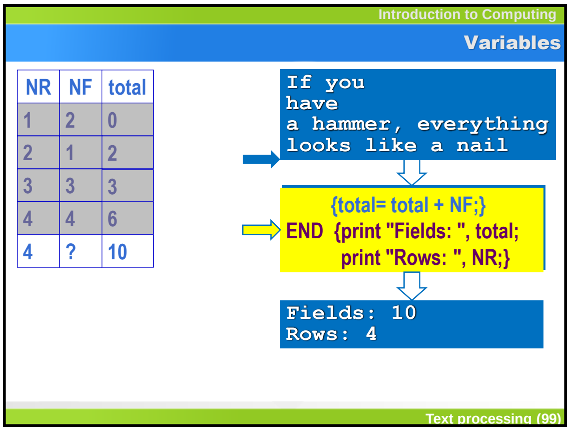
Variables

NR	NF	total
1	2	0
2	1	2
3	3	3

```
If you
have
a hammer, everything
looks like a nail
```

```
{total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

Text processing (98)



Introduction to Computing

Variables

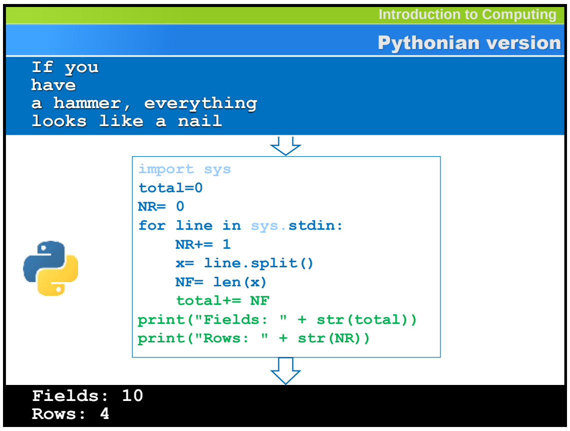
NR	NF	total
1	2	0
2	1	2
3	3	3
4	4	6
4	?	10

```
If you
have
a hammer, everything
looks like a nail
```

```
{total= total + NF;}
END {print "Fields: ", total;
     print "Rows: ", NR;}
```

```
Fields: 10
Rows: 4
```

Text processing (99)



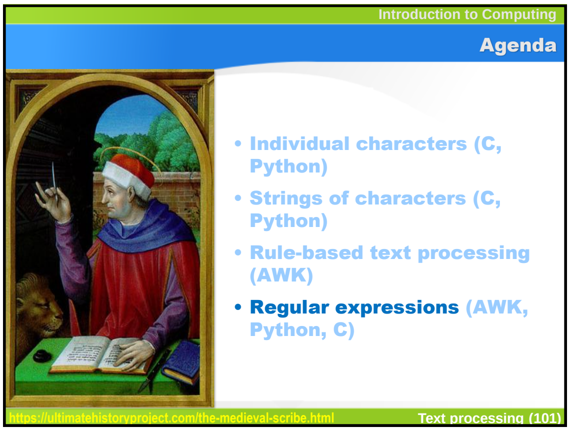
Introduction to Computing

Pythonian version

```
If you
have
a hammer, everything
looks like a nail
```

```
import sys
total=0
NR= 0
for line in sys.stdin:
    NR+= 1
    x= line.split()
    NF= len(x)
    total+= NF
print("Fields: " + str(total))
print("Rows: " + str(NR))
```

```
Fields: 10
Rows: 4
```



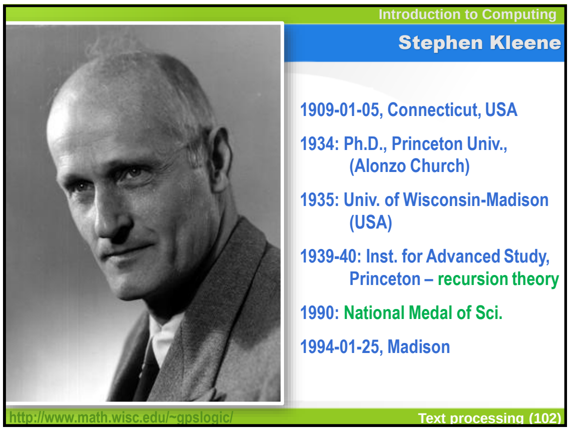
Introduction to Computing

Agenda

- Individual characters (C, Python)
- Strings of characters (C, Python)
- Rule-based text processing (AWK)
- **Regular expressions** (AWK, Python, C)

https://ultimatehistoryproject.com/the-medieval-scribe.html

Text processing (101)



Introduction to Computing

Stephen Kleene

1909-01-05, Connecticut, USA

1934: Ph.D., Princeton Univ., (Alonzo Church)

1935: Univ. of Wisconsin-Madison (USA)

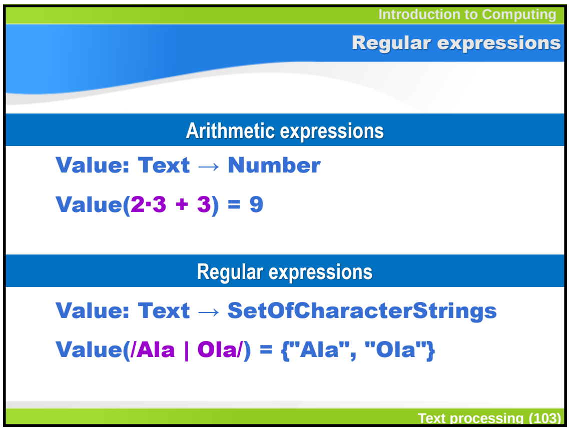
1939-40: Inst. for Advanced Study, Princeton – recursion theory

1990: National Medal of Sci.

1994-01-25, Madison

http://www.math.wisc.edu/~gpslogic/

Text processing (102)



Introduction to Computing

Regular expressions

Arithmetic expressions

Value: Text → Number

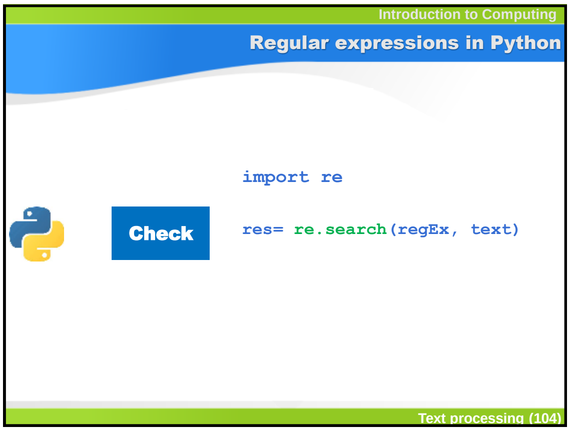
Value(2·3 + 3) = 9

Regular expressions

Value: Text → SetOfCharacterStrings

Value(/Ala | Ola/) = {"Ala", "Ola"}

Text processing (103)



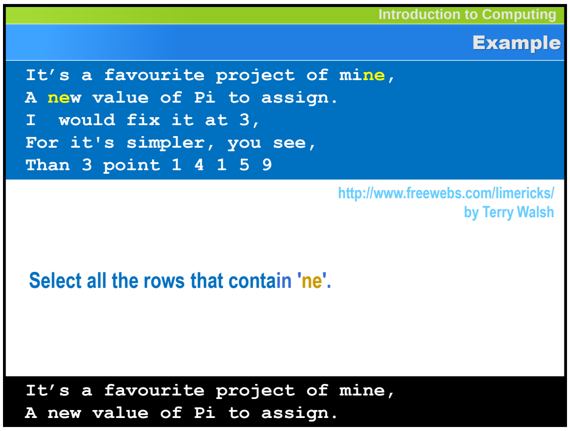
Introduction to Computing

Regular expressions in Python

Check

```
import re

res= re.search(regEx, text)
```

Text processing (104)



Introduction to Computing

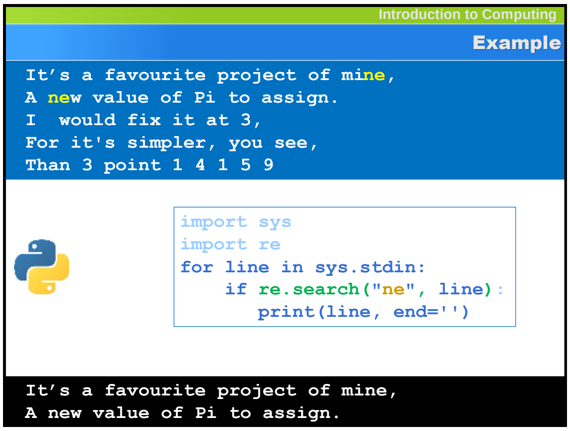
Example

```
It’s a favourite project of mine,
A new value of Pi to assign.
I  would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9
```

http://www.freewebs.com/limericks/
by Terry Walsh

Select all the rows that contain 'ne'.

```
It’s a favourite project of mine,
A new value of Pi to assign.
```



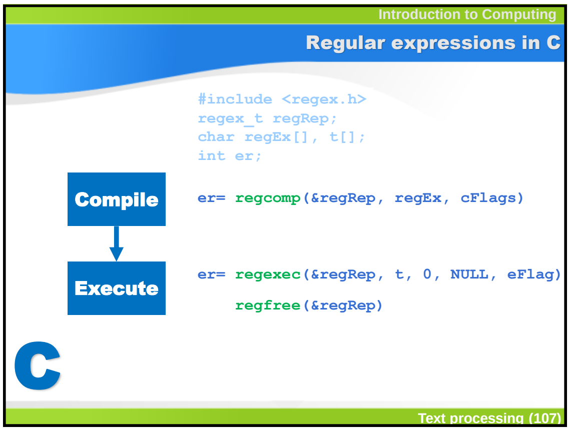
Introduction to Computing

Example

```
It’s a favourite project of mine,
A new value of Pi to assign.
I  would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9
```

```
import sys
import re
for line in sys.stdin:
    if re.search("ne", line):
       print(line, end='')
```

```
It’s a favourite project of mine,
A new value of Pi to assign.
```



Introduction to Computing

Regular expressions in C

```
#include <regex.h>
regex_t regRep;
char regEx[], t[];
int er;
```

Compile

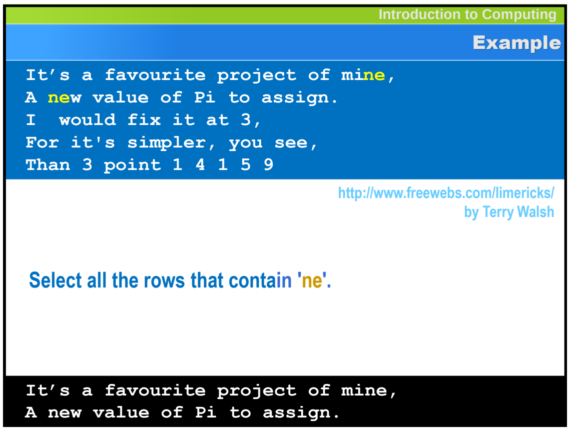
```
er= regcomp(&regRep, regEx, cFlags)
```

Execute

```
er= regexec(&regRep, t, 0, NULL, eFlag)
    regfree(&regRep)
```

C

Text processing (107)



Introduction to Computing

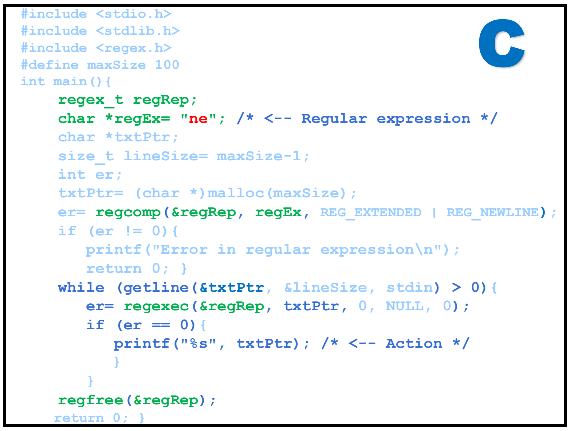
Example

```
It’s a favourite project of mine,
A new value of Pi to assign.
I  would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9
```

http://www.freewebs.com/limericks/
by Terry Walsh

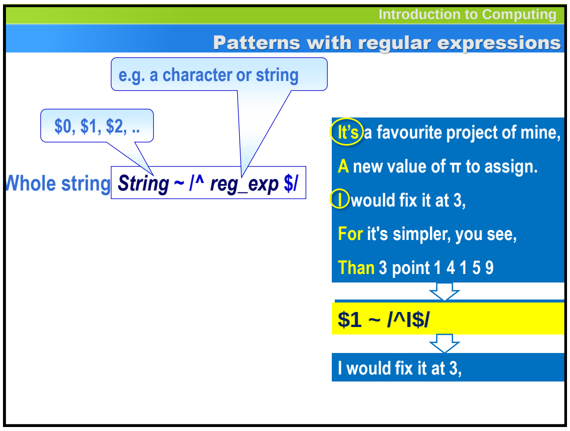
Select all the rows that contain 'ne'.

```
It’s a favourite project of mine,
A new value of Pi to assign.
```



C

```
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#define maxSize 100
int main(){
    regex_t regRep;
    char *regEx= "ne"; /* <-- Regular expression */
    char *txtPtr;
    size_t lineSize= maxSize-1;
    int er;
    txtPtr= (char *)malloc(maxSize);
    er= regcomp(&regRep, regEx, REG_EXTENDED | REG_NEWLINE);
    if (er != 0){
       printf("Error in regular expression\n");
       return 0; }
    while (getline(&txtPtr, &lineSize, stdin) > 0){
       er= regexec(&regRep, txtPtr, 0, NULL, 0);
       if (er == 0){
          printf("%s", txtPtr); /* <-- Action */
          }
       }
    regfree(&regRep);
   return 0; }
```



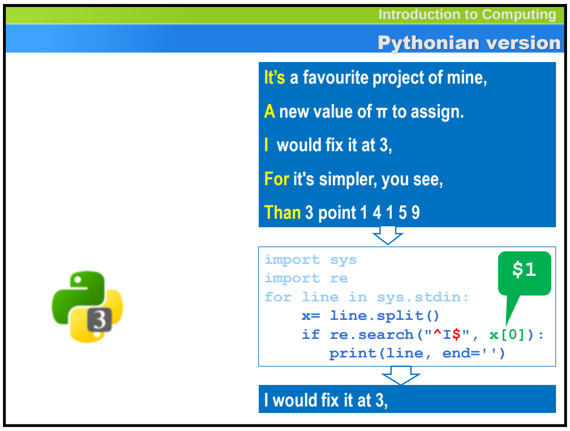
Introduction to Computing

Patterns with regular expressions

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

$1 ~ /^I$/

I would fix it at 3,



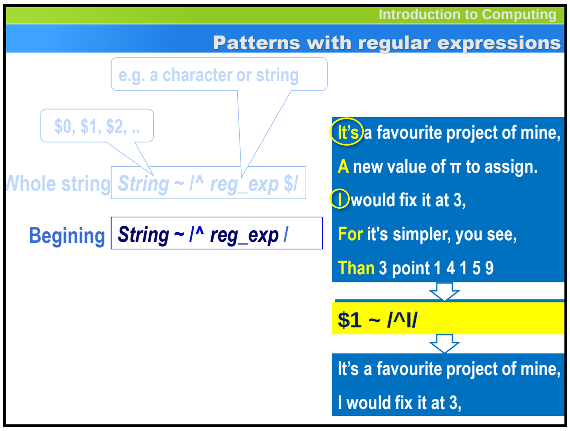
Introduction to Computing

Pythonian version

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

```
import sys
import re
for line in sys.stdin:
    x= line.split()
    if re.search("^I$", x[0]):
       print(line, end='')
```

I would fix it at 3,



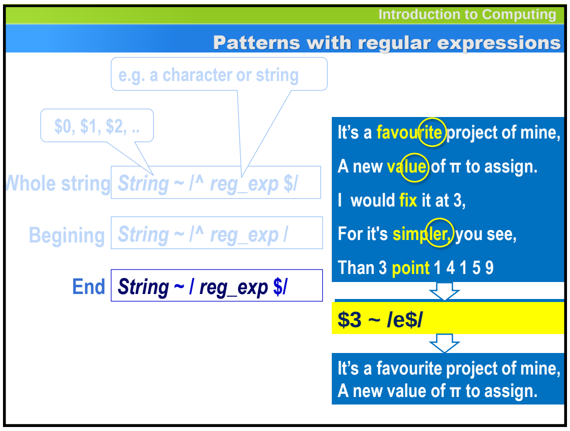
Introduction to Computing

Patterns with regular expressions

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

$1 ~ /^I/

It's a favourite project of mine,
I would fix it at 3,



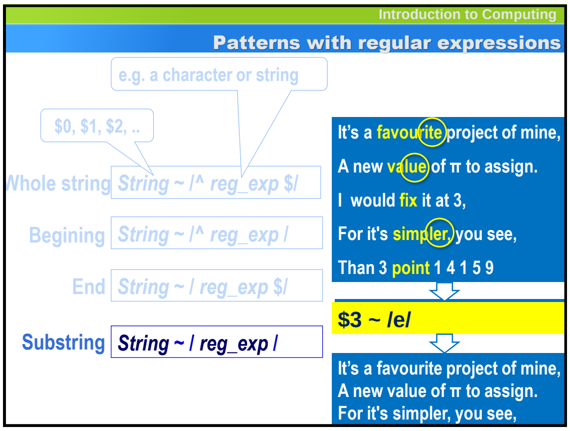
Introduction to Computing

Patterns with regular expressions

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

$3 ~ /e$/

It's a favourite project of mine,
A new value of π to assign.



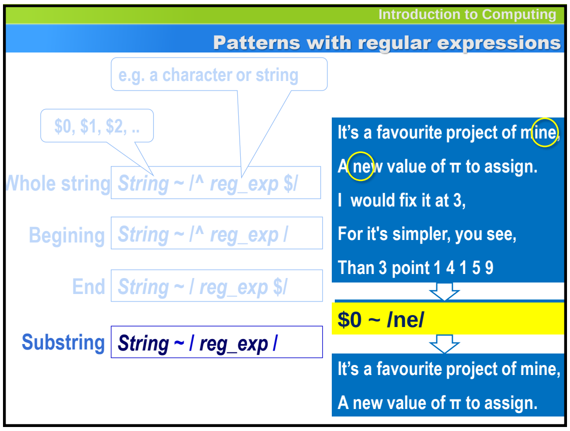
Introduction to Computing

Patterns with regular expressions

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

$3 ~ /e/

It's a favourite project of mine,
A new value of π to assign.
For it's simpler, you see,



Introduction to Computing

Patterns with regular expressions

e.g. a character or string

$0, $1, $2, ..

Whole string String ~ /^ reg_exp $/

Begining String ~ /^ reg_exp /

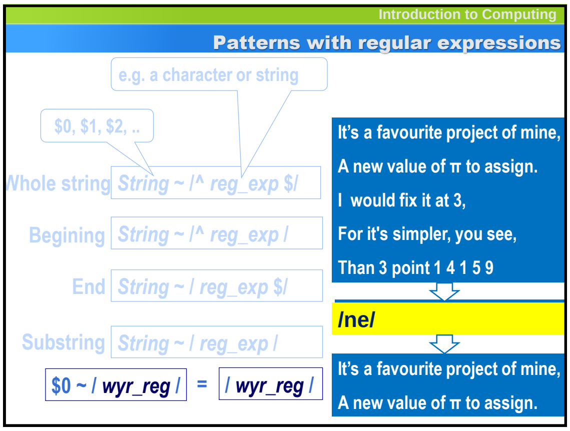
End String ~ / reg_exp $/

Substring **String ~ / reg_exp /**

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

$0 ~ /ne/

It's a favourite project of mine,
A new value of π to assign.



Introduction to Computing

Patterns with regular expressions

e.g. a character or string

$0, $1, $2, ..

Whole string String ~ /^ reg_exp $/

Begining String ~ /^ reg_exp /

End String ~ / reg_exp $/

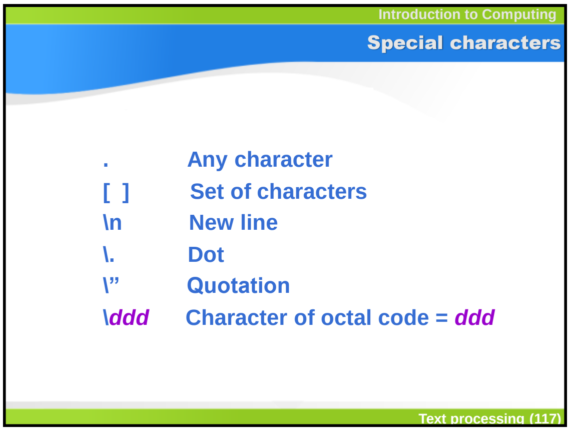
Substring String ~ / reg_exp /

$0 ~ / wyr_reg / = / wyr_reg /

It's a favourite project of mine,
A new value of π to assign.
I would fix it at 3,
For it's simpler, you see,
Than 3 point 1 4 1 5 9

/ne/

It's a favourite project of mine,
A new value of π to assign.

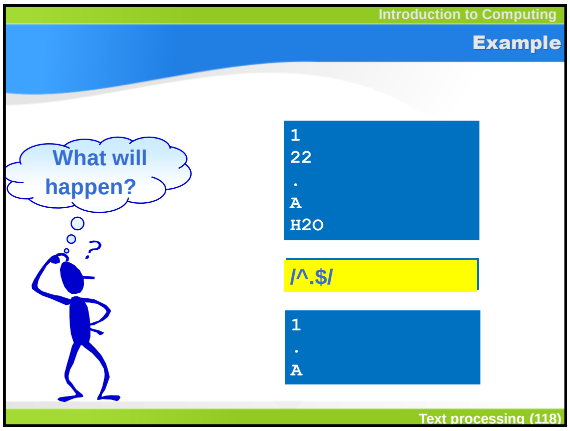


Introduction to Computing

Special characters

.	Any character
[ ]	Set of characters
\n	New line
\.	Dot
\"	Quotation
ddd	Character of octal code = *ddd*

Text processing (117)



Introduction to Computing

Example

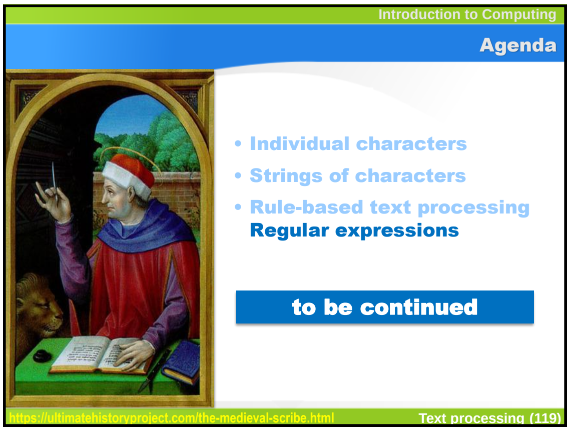
What will happen?

```
1
22
.
A
H2O
```

/^.$/

```
1
.
A
```

Text processing (118)



Introduction to Computing

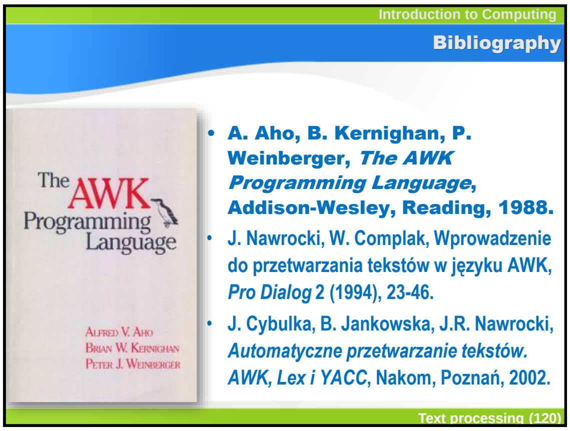
Agenda

- Individual characters
- Strings of characters
- Rule-based text processing
  Regular expressions

to be continued

https://ultimatehistoryproject.com/the-medieval-scribe.html

Text processing (119)



Introduction to Computing

Bibliography

The AWK Programming Language
Alfred V. Aho
Brian W. Kernighan
Peter J. Weinberger

- **A. Aho, B. Kernighan, P. Weinberger, *The AWK Programming Language*, Addison-Wesley, Reading, 1988.**
- J. Nawrocki, W. Complak, Wprowadzenie do przetwarzania tekstów w języku AWK, *Pro Dialog* 2 (1994), 23-46.
- J. Cybulka, B. Jankowska, J.R. Nawrocki, *Automatyczne przetwarzanie tekstów. AWK, Lex i YACC*, Nakom, Poznań, 2002.

Text processing (120)