



# Advanced Linux Usage

2019-11-21

Anders Sjölander  
anders.sjolander@uppmax.uu.se

Enabler for Life Sciences

- Same program, many files

```
$ ls -l
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_9.bam
$ my_prog sample_1.bam
```

- Same program, many files

```
$ ls -l
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_9.bam
$ my_prog sample_1.bam
$ my_prog sample_2.bam
```

- Same program, many files

```
$ ls -l
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_9.bam
$ my_prog sample_1.bam
$ my_prog sample_2.bam
$ my_prog sample_3.bam
$ my_prog sample_4.bam
$ my_prog sample_5.bam
$ my_prog sample_6.bam
$ my_prog sample_7.bam
$ my_prog sample_8.bam
$ my_prog sample_9.bam
$
```

# Multiple files

- Same program, many files
  - 10 files? Ok
  - 1000 files? Not ok

# Multiple files

- Same program, many files
  - 10 files? Ok
  - 1000 files? Not ok
- Reproducibility
  - Self and others

# Multiple files

- Same program, many files
  - 10 files? Ok
  - 1000 files? Not ok
- Reproducibility
  - Self and others

A solution - write a script!



# Basic script

```
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_9.bam
$ nano analysis.sh
```

GNU nano 2.0.9

File: analysis.sh

Modified

**^G** Get Help  
**^X** Exit

**^O** WriteOut  
**^J** Justify

**^R** Read File  
**^W** Where Is

**^Y** Prev Page  
**^V** Next Page

**^K** Cut Text  
**^U** UnCut Text

**^C** Cur Pos  
**^T** To Spell

GNU nano 2.0.9

File: analysis.sh

Modified

my\_prog sample\_1.bam

**^G** Get Help  
**^X** Exit

**^O** WriteOut  
**^J** Justify

**^R** Read File  
**^W** Where Is

**^Y** Prev Page  
**^V** Next Page

**^K** Cut Text  
**^U** UnCut Text

**^C** Cur Pos  
**^T** To Spell

GNU nano 2.0.9

File: analysis.sh

Modified

```
my_prog sample_1.bam  
my_prog sample_2.bam
```

**^G** Get Help  
**^X** Exit

**^O** WriteOut  
**^J** Justify

**^R** Read File  
**^W** Where Is

**^Y** Prev Page  
**^V** Next Page

**^K** Cut Text  
**^U** UnCut Text

**^C** Cur Pos  
**^T** To Spell

GNU nano 2.0.9

File: analysis.sh

Modified

```
my_prog sample_1.bam
my_prog sample_2.bam
my_prog sample_3.bam
my_prog sample_4.bam
my_prog sample_5.bam
my_prog sample_6.bam
my_prog sample_7.bam
my_prog sample_8.bam
my_prog sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** WriteOut  
**^J** Justify

**^R** Read File  
**^W** Where Is

**^Y** Prev Page  
**^V** Next Page

**^K** Cut Text  
**^U** UnCut Text

**^C** Cur Pos  
**^T** To Spell

# Basic script

```
$ l
total 4,0K
-rw-rw-r-- 1 dahlo dahlo 267 Sep  7 09:34 analysis.sh
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_9.bam
$
```

# Basic script

```
$ l
total 4,0K
-rw-rw-r-- 1 dahlo dahlo 267 Sep  7 09:34 analysis.sh
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_9.bam
$ bash analysis.sh
```

```
$ l
total 4,0K
-rw-rw-r-- 1 dahlo dahlo 267 Sep  7 09:34 analysis.sh
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_9.bam
$ bash analysis.sh
```

Still not OK for 1000 or more files!



GNU nano 2.0.9

File: analysis.sh

Modified

```
my_prog sample_1.bam
my_prog sample_2.bam
my_prog sample_3.bam
my_prog sample_4.bam
my_prog sample_5.bam
my_prog sample_6.bam
my_prog sample_7.bam
my_prog sample_8.bam
my_prog sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** WriteOut  
**^J** Justify

**^R** Read File  
**^W** Where Is

**^Y** Prev Page  
**^V** Next Page

**^K** Cut Text  
**^U** UnCut Text

**^C** Cur Pos  
**^T** To Spell

GNU nano 2.5.3

File: analysis.sh

```
my_prog -r references/human_genome.fa sample_1.bam
my_prog -r references/human_genome.fa sample_2.bam
my_prog -r references/human_genome.fa sample_3.bam
my_prog -r references/human_genome.fa sample_4.bam
my_prog -r references/human_genome.fa sample_5.bam
my_prog -r references/human_genome.fa sample_6.bam
my_prog -r references/human_genome.fa sample_7.bam
my_prog -r references/human_genome.fa sample_8.bam
my_prog -r references/human_genome.fa sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Assigning**  
`my_variable=5`  
`my_variable="nice text"`
- **Using**  
`$my_variable`

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Using**

```
$my_variable
```

```
$ my_variable="Dave"
```

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Using**

```
$my_variable
```

```
$ my_variable="Dave"
```

```
$ echo "Hello, $my_variable."
```

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Using**

```
$my_variable
```

```
$ my_variable="Dave"
```

```
$ echo "Hello, $my_variable."
```

```
Hello, Dave.
```

GNU nano 2.5.3

File: analysis.sh

```
my_prog -r references/human_genome.fa sample_1.bam
my_prog -r references/human_genome.fa sample_2.bam
my_prog -r references/human_genome.fa sample_3.bam
my_prog -r references/human_genome.fa sample_4.bam
my_prog -r references/human_genome.fa sample_5.bam
my_prog -r references/human_genome.fa sample_6.bam
my_prog -r references/human_genome.fa sample_7.bam
my_prog -r references/human_genome.fa sample_8.bam
my_prog -r references/human_genome.fa sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page



GNU nano 2.5.3

File: analysis.sh

```
ref=references/human_genome.fa
```

```
my_prog -r $ref sample_1.bam
my_prog -r $ref sample_2.bam
my_prog -r $ref sample_3.bam
my_prog -r $ref sample_4.bam
my_prog -r $ref sample_5.bam
my_prog -r $ref sample_6.bam
my_prog -r $ref sample_7.bam
my_prog -r $ref sample_8.bam
my_prog -r $ref sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\** Replace

[ Read 12 lines ]

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
```

```
my_prog -r $ref sample_1.bam  
my_prog -r $ref sample_2.bam  
my_prog -r $ref sample_3.bam  
my_prog -r $ref sample_4.bam  
my_prog -r $ref sample_5.bam  
my_prog -r $ref sample_6.bam  
my_prog -r $ref sample_7.bam  
my_prog -r $ref sample_8.bam  
my_prog -r $ref sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

```
for var in 1 2 3;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
1  
2  
3  
$
```

```
for var in text works too;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
text  
works  
too  
$
```

```
for var in mix them 5;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
mix  
them  
5  
$
```

```
for var in *.txt;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
all.txt  
examples.txt  
readme.txt
```

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
```

```
my_prog -r $ref sample_1.bam  
my_prog -r $ref sample_2.bam  
my_prog -r $ref sample_3.bam  
my_prog -r $ref sample_4.bam  
my_prog -r $ref sample_5.bam  
my_prog -r $ref sample_6.bam  
my_prog -r $ref sample_7.bam  
my_prog -r $ref sample_8.bam  
my_prog -r $ref sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in *.bam ;
do
    my_prog -r $ref $file
done
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page



GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in *.bam ;
do
    echo my_prog -r $ref $file
done
```

## Debugging!

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in *.bam ;
do
    echo my_prog -r $ref $file
done
```

```
$ bash analysis.sh
my_prog -r references/goat_genome_version4.1.fa sample_1.bam
my_prog -r references/goat_genome_version4.1.fa sample_2.bam
my_prog -r references/goat_genome_version4.1.fa sample_3.bam
my_prog -r references/goat_genome_version4.1.fa sample_4.bam
my_prog -r references/goat_genome_version4.1.fa sample_5.bam
my_prog -r references/goat_genome_version4.1.fa sample_6.bam
my_prog -r references/goat_genome_version4.1.fa sample_7.bam
my_prog -r references/goat_genome_version4.1.fa sample_8.bam
my_prog -r references/goat_genome_version4.1.fa sample_9.bam
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_** Replace

**^K** Cut Text  
**^U** Uncut Te

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in *.bam ;
do
    my_prog -r $ref $file
done
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

# Basic script

```
$ bash analysis.sh
```

# Basic script

```
$ bash analysis.sh data/
```

# Basic script

```
$ bash analysis.sh data/
```

\$1

```
$ bash analysis.sh data/ second_argument
```

\$1

\$2

```
$ bash analysis.sh data/ second_argument third
```

\$1

\$2

\$3



# Basic script

```
$ bash analysis.sh data/ second_argument third "fourth argument"
```

\$1

\$2

\$3

\$4

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa  
for file in *.bam ;  
do  
    my_prog -r $ref $file  
done
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in $1/*.bam ;
do
    my_prog -r $ref $file
done
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

# Basic script

```
$ cat file.list  
sample_1.bam  
sample_3.bam  
smample_9.bam
```

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in $1/*.bam ;
do
    my_prog -r $ref $file
done
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in $( cat $1 ) ;
do
    my_prog -r $ref $file
done
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_\** Replace

**^K** Cut Text  
**^U** Uncut Text

**^J** Justify  
**^T** To Spell

**^C** Cur Pos  
**^\_** Go To Line

**^Y** Prev Page  
**^V** Next Page

# Basic script

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
for file in $( cat $1 ) ;
do
    my_prog -r $ref $file
done
```

```
$ cat file.list
sample_1.bam
sample_3.bam
sample_9.bam
$ bash analysis.sh file.list
```

**^G** Get Help  
**^X** Exit

**^O** Write Out  
**^R** Read File

**^W** Where Is  
**^\_** Replace

**^K** Cut Text  
**^U** Uncut Text

**^**  
**^**

- Control statement

```
if condition; then  
    action  
fi
```



- Control statement

```
if true; then  
    echo "This is true"  
fi
```

result:  
This is true

- Control statement

```
if false; then  
    echo "This is true"  
fi
```

result:

- Control statement

```
if [[ 5 < 9 ]]; then  
    echo "This is true"  
fi
```

result:  
This is true

- Control statement

```
if [[ 5 > 9 ]]; then  
    echo "This is true"  
fi
```

result:

- Control statement

```
if [[ 5 == 9 ]]; then  
    echo "This is true"  
fi
```

result:

- Control statement

```
if [[ "Hello" == "Hello" ]]; then  
    echo "This is true"  
fi
```

result:  
This is true

- Control statement

```
if [[ "Hello" == "Hi" ]]; then  
    echo "This is true"  
fi
```

result:

- Control statement

```
if [[ "Hello" == "Hel"* ]]; then  
    echo "This is true"  
fi
```

result:  
This is true



- For all samples except dog

```
for file in $1/*.bam ;  
do  
    echo my_prog $file  
done
```

- For all samples except dog

```
for file in $1/*.bam ;  
do  
    if [[ ... != "dog"* ]]; then  
        echo my_prog $file  
    fi  
done
```

- For all samples except dog

```
for file in $1/*.bam ;  
do  
    if [[ ... != "dog"* ]]; then  
        echo my_prog $file  
    fi  
done
```

Ex: \$file is /path/to/dog\_1.bam

- For all samples except dog

```
for file in $1/*.bam ;
do
    if [[ ... != "dog"* ]]; then
        echo my_prog $file
    fi
done
```

Ex: \$file is /path/to/dog\_1.bam

basename \$file

- For all samples except dog

```
for file in $1/*.bam ;
do
    if [[ ... != "dog"* ]]; then
        echo my_prog $file
    fi
done
```

Ex: \$file is /path/to/dog\_1.bam

basename \$file

dog\_1.bam

- For all samples except dog

```
for file in $1/*.bam ;
do
    if [[ $(basename $file) != "dog"* ]]; then
        echo my_prog $file
    fi
done
```

Ex: \$file is /path/to/dog\_1.bam

basename \$file

dog\_1.bam

- For all samples except dog

```
for file in $1/*.bam ;
do
    if [[ $(basename $file) != "dog"* ]]; then
        my_prog $file
    fi
done
```

Ex: \$file is /path/to/dog\_1.bam

basename \$file

dog\_1.bam

# Different languages

- Programming is programming
  - Perl, Python, Bash, and more



- Programming is programming
  - Perl, Python, **Bash**, and more

```
for file in $1/*.bam ;  
do  
    if [[ $(basename $file) != "dog"* ]]; then  
        my_prog $file  
    fi  
done
```

# Different languages

- Programming is programming
  - **Perl**, Python, Bash, and more

```
for file in $1/*.bam ;  
do  
    if [[ $(basename $file) != "dog"* ]]; then  
        my_prog $file  
    fi  
done
```

```
use strict;  
use warnings;  
use File::Basename;  
  
foreach my $file (glob("$ARGV[0]/*.bam")) {  
    if(basename($file) !~ "^dog.+"){  
        system("my_prog", $file);  
    }  
}
```

# Different languages

- Programming is programming
  - Perl, **Python**, Bash, and more

```
for file in $1/*.bam ;  
do  
    if [[ $(basename $file) != "dog"* ]]; then  
        my_prog $file  
    fi  
done
```

```
import glob  
import sys  
import subprocess  
import os  
  
for file in glob.glob( sys.argv[1] + "/*.bam" ):  
    if not os.path.basename(file).startswith("dog"):  
        subprocess.call( ["my_prog" , file] )
```

# Different languages

- Programming is programming
  - Perl, Python, Bash, and more
- Start with one, git gud, (learn another)

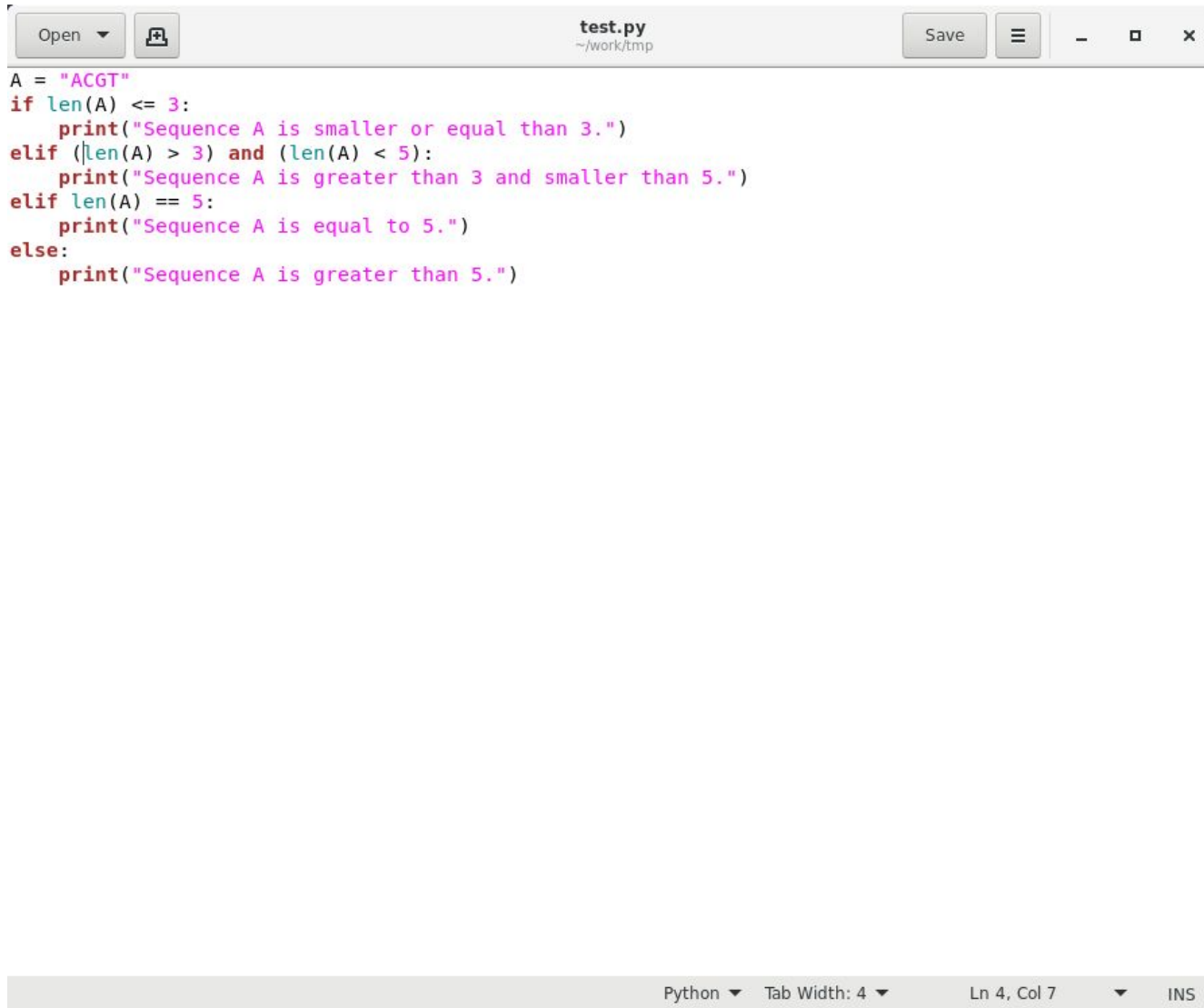
# Different languages

- Programming is programming
  - Perl, Python, Bash, and more
- Start with one, git gud, (learn another)

PYTHON

- Graphical text editor more similar to what you might be used to
- Launch through command line:

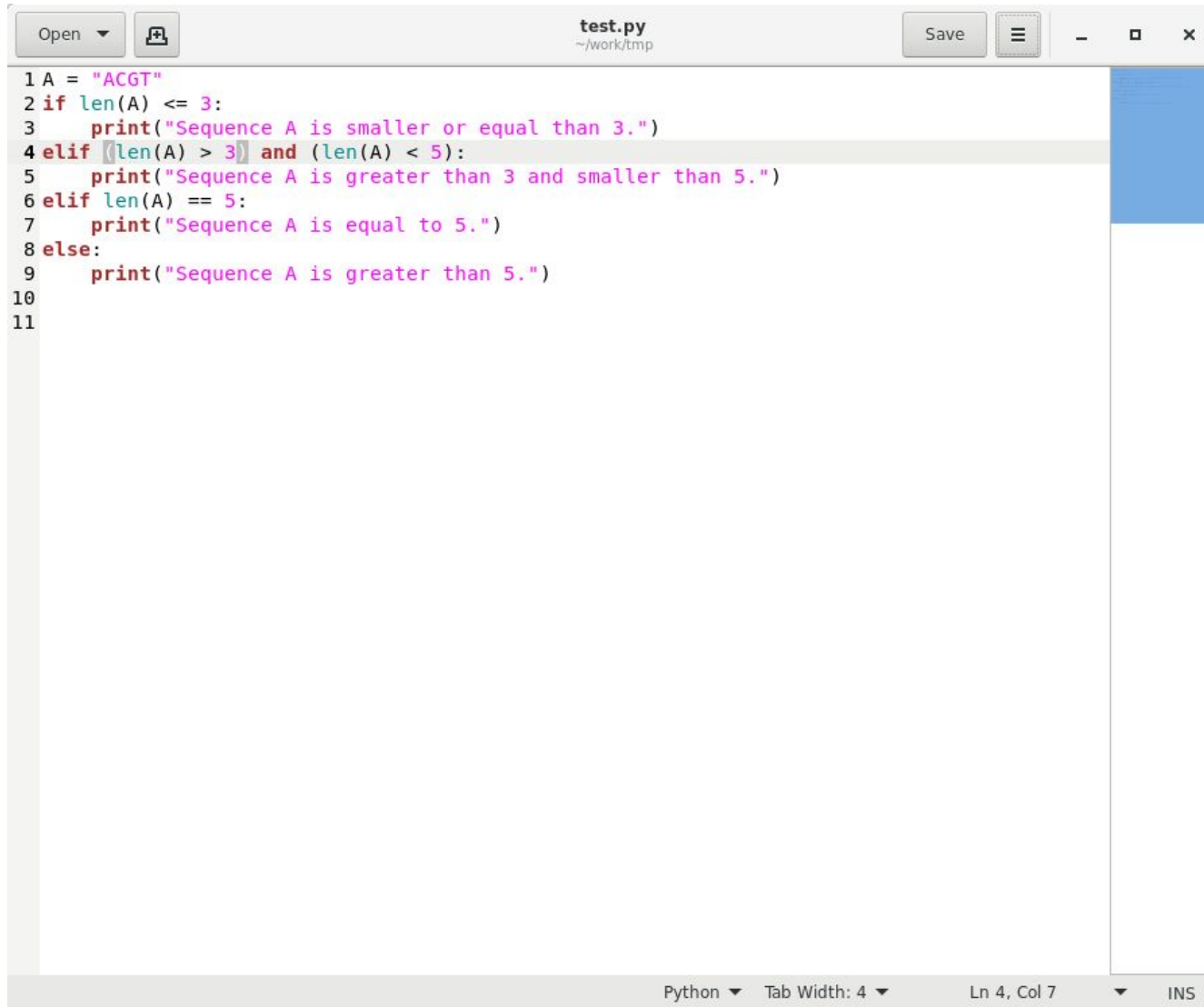
```
$ gedit
```



The image shows a screenshot of the Gedit text editor. The window title bar at the top displays "test.py" and the file path "~/.work/tmp". On the left side of the title bar are buttons for "Open" and a file icon. On the right side are buttons for "Save", a menu icon (three horizontal lines), and window control icons (minimize, maximize, and close). The main editing area contains a Python script with the following code:

```
A = "ACGT"
if len(A) <= 3:
    print("Sequence A is smaller or equal than 3.")
elif (len(A) > 3) and (len(A) < 5):
    print("Sequence A is greater than 3 and smaller than 5.")
elif len(A) == 5:
    print("Sequence A is equal to 5.")
else:
    print("Sequence A is greater than 5.")
```

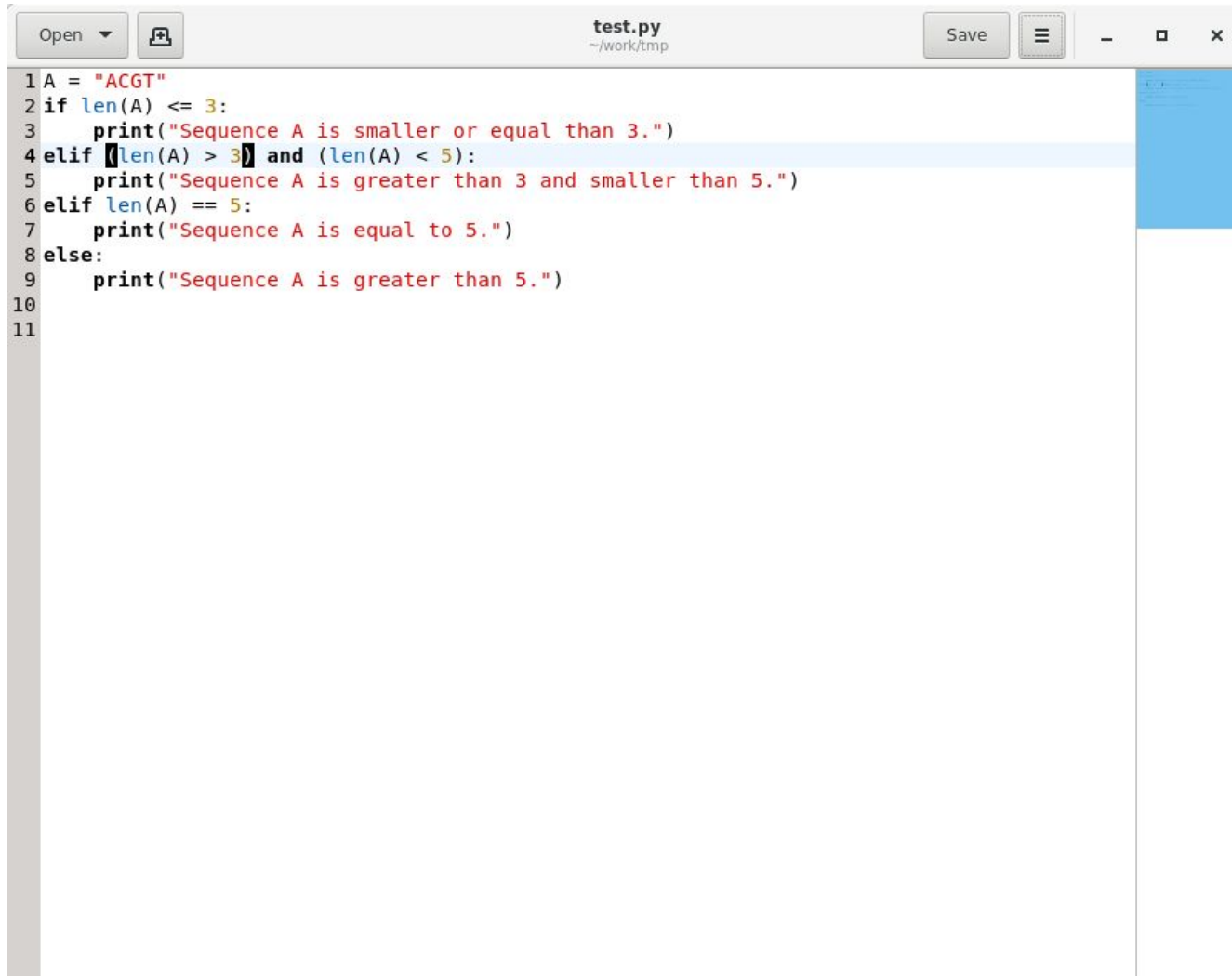
The status bar at the bottom of the window shows "Python" with a dropdown arrow, "Tab Width: 4" with a dropdown arrow, "Ln 4, Col 7" with a dropdown arrow, and "INS" in the final mode indicator.



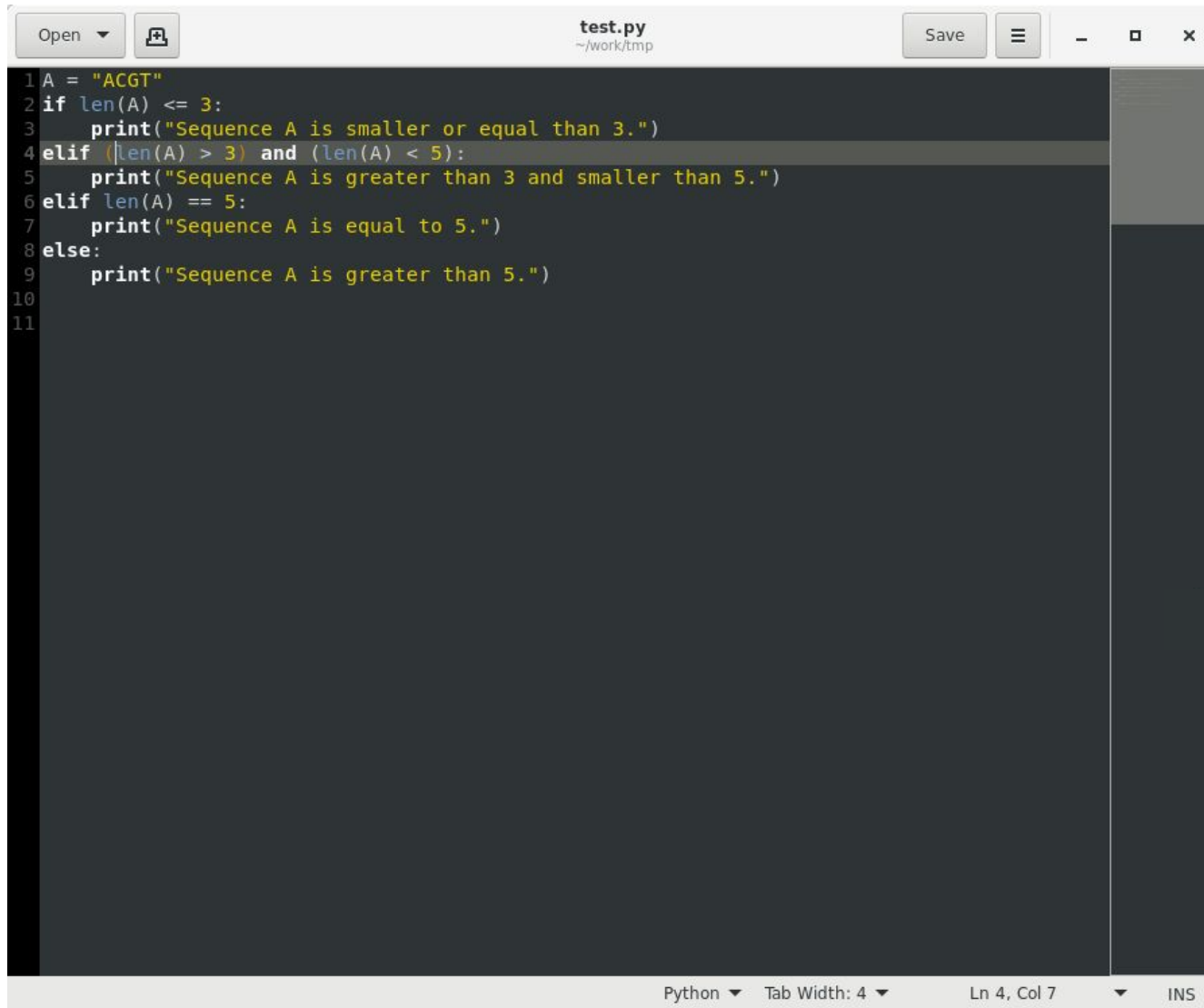
```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif len(A) > 3 and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```

Python Tab Width: 4 Ln 4, Col 7 INS



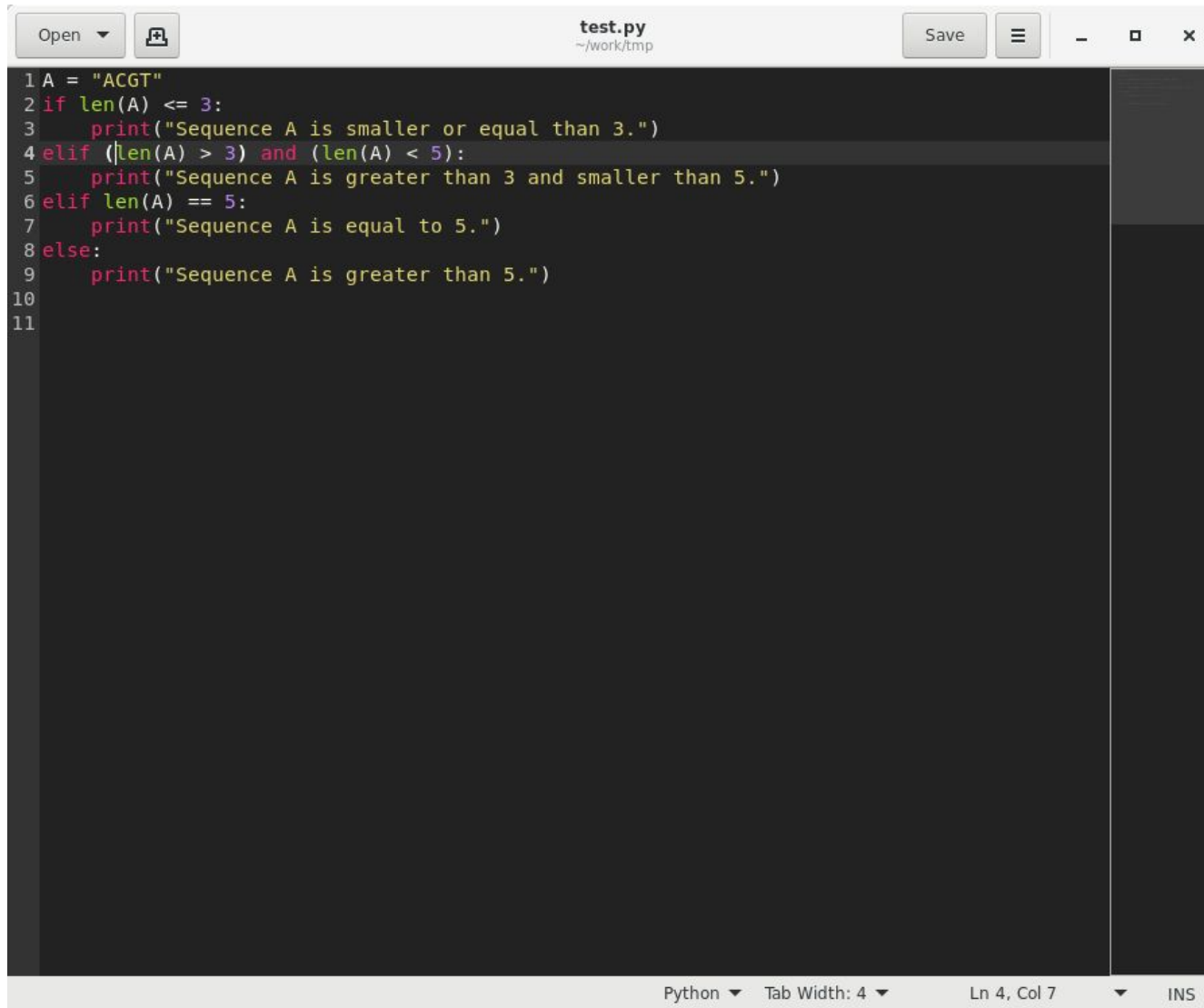


```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif len(A) > 3 and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```



```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif (len(A) > 3) and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```

Python Tab Width: 4 Ln 4, Col 7 INS



```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif (len(A) > 3) and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```

Python Tab Width: 4 Ln 4, Col 7 INS

- Menu - Preferences - View
  - Display line numbers
  - Display overview map
  - Highlight current line
  - Highlight matching brackets
- Menu - Preferences - Editor
  - Tab width 4
  - Insert spaces instead of tabs
- Menu - Preferences - Fonts & Colors
  - Kate or Oblivion (or Monokai installed manually, google it)

Laboratory time once again!

[https://scilifelab.github.io/courses/ngsintro/1911/labs/loops\\_lab](https://scilifelab.github.io/courses/ngsintro/1911/labs/loops_lab)