

Data Science with Python and R

Moinul Islam

9/15/2022

Table of contents

Preface	7
1 Basic Python	8
1.1 Prerequisites	8
1.2 Variables	8
1.3 Strings	8
1.3.1 Change case of string	9
1.4 Numbers	9
1.5 Comments	11
1.6 Lists	11
1.6.1 Neat outcome	12
1.6.2 Print string	12
1.7 Functions	13
2 Introduction to Python	14
2.1 Prerequisites	14
2.2 Variables	14
2.3 Strings	14
2.3.1 Change case of string	15
2.4 Numbers	15
2.5 Comments	17
2.6 Lists	17
2.6.1 Neat outcome	18
2.6.2 Print string	18
2.7 Functions	19
3 Introduction to Python	20
3.1 Prerequisites	20
3.2 Variables	20
3.3 Strings	20
3.3.1 Change case of string	21
3.4 Numbers	21
3.5 Comments	23
3.6 Lists	23
3.6.1 Neat outcome	24

3.6.2	Print string	24
3.7	Functions	25
4	Introduction to Python	26
4.1	Prerequisites	26
4.2	Variables	26
4.3	Strings	26
4.3.1	Change case of string	27
4.4	Numbers	27
4.5	Comments	29
4.6	Lists	29
4.6.1	Neat outcome	30
4.6.2	Print string	30
4.7	Functions	31
5	Introduction to Python	32
5.1	Prerequisites	32
5.2	Variables	32
5.3	Strings	32
5.3.1	Change case of string	33
5.4	Numbers	33
5.5	Comments	35
5.6	Lists	35
5.6.1	Neat outcome	36
5.6.2	Print string	36
5.7	Functions	37
6	Introduction to Python	38
6.1	Prerequisites	38
6.2	Variables	38
6.3	Strings	38
6.3.1	Change case of string	39
6.4	Numbers	39
6.5	Comments	41
6.6	Lists	41
6.6.1	Neat outcome	42
6.6.2	Print string	42
6.7	Functions	43
7	Introduction to Python	44
7.1	Prerequisites	44
7.2	Variables	44

7.3	Strings	44
7.3.1	Change case of string	45
7.4	Numbers	45
7.5	Comments	47
7.6	Lists	47
7.6.1	Neat outcome	48
7.6.2	Print string	48
7.7	Functions	49
8	Introduction to Python	50
8.1	Prerequisites	50
8.2	Variables	50
8.3	Strings	50
8.3.1	Change case of string	51
8.4	Numbers	51
8.5	Comments	53
8.6	Lists	53
8.6.1	Neat outcome	54
8.6.2	Print string	54
8.7	Functions	55
9	Introduction to Python	56
9.1	Prerequisites	56
9.2	Variables	56
9.3	Strings	56
9.3.1	Change case of string	57
9.4	Numbers	57
9.5	Comments	59
9.6	Lists	59
9.6.1	Neat outcome	60
9.6.2	Print string	60
9.7	Functions	61
10	Introduction to Python	62
10.1	Prerequisites	62
10.2	Variables	62
10.3	Strings	62
10.3.1	Change case of string	63
10.4	Numbers	63
10.5	Comments	65
10.6	Lists	65
10.6.1	Neat outcome	66
10.6.2	Print string	66

10.7 Functions	67
11 Introduction to Python	68
11.1 Prerequisites	68
11.2 Variables	68
11.3 Strings	68
11.3.1 Change case of string	69
11.4 Numbers	69
11.5 Comments	71
11.6 Lists	71
11.6.1 Neat outcome	72
11.6.2 Print string	72
11.7 Functions	73
12 Introduction to Python	74
12.1 Prerequisites	74
12.2 Variables	74
12.3 Strings	74
12.3.1 Change case of string	75
12.4 Numbers	75
12.5 Comments	77
12.6 Lists	77
12.6.1 Neat outcome	78
12.6.2 Print string	78
12.7 Functions	79
13 Introduction to Python	80
13.1 Prerequisites	80
13.2 Variables	80
13.3 Strings	80
13.3.1 Change case of string	81
13.4 Numbers	81
13.5 Comments	83
13.6 Lists	83
13.6.1 Neat outcome	84
13.6.2 Print string	84
13.7 Functions	85
14 Introduction to Python	86
14.1 Prerequisites	86
14.2 Variables	86
14.3 Strings	86
14.3.1 Change case of string	87

14.4	Numbers	87
14.5	Comments	89
14.6	Lists	89
14.6.1	Neat outcome	90
14.6.2	Print string	90
14.7	Functions	91
15	Introduction to Python	92
15.1	Prerequisites	92
15.2	Variables	92
15.3	Strings	92
15.3.1	Change case of string	93
15.4	Numbers	93
15.5	Comments	95
15.6	Lists	95
15.6.1	Neat outcome	96
15.6.2	Print string	96
15.7	Functions	97
	References	98

Preface

This book contains the basic Python and R programming skills to learn data science. You can dive into the amazing data science world by using Python and R. Learn how to effectively analyze and visualize your data. No coding experience or skills needed. Data Science is about data gathering, analysis and decision-making. Data Science is used in many industries in the world today, e.g. banking, consultancy, healthcare, and manufacturing. Data science is not a one-step process such that will get to learned it in a short time. It's passes from many stages and every element is important. One should always follow the proper steps to reach the ladder. Every step has its value and it counts in your model.



Figure 1: Data Science

1 Basic Python

1.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

1.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

1.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language


```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

1.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

1.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

1.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

1.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

1.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

1.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

1.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

2 Introduction to Python

2.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

2.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

2.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

2.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

2.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```


14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

2.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

2.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

2.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

2.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

2.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

3 Introduction to Python

3.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

3.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

3.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

3.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

3.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

3.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

3.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

3.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

3.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```


3.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

4 Introduction to Python

4.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

4.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

4.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

4.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

4.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

4.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

4.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

4.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

4.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

4.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

5 Introduction to Python

5.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

5.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

5.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language


```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

5.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

5.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

5.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

5.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

5.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

5.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

5.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

6 Introduction to Python

6.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

6.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

6.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

6.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

6.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```


14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

6.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

6.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

6.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

6.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

6.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

7 Introduction to Python

7.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

7.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

7.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

7.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

7.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

7.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

7.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

7.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

7.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```


7.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

8 Introduction to Python

8.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

8.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

8.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

8.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

8.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

8.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

8.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

8.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

8.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

8.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

9 Introduction to Python

9.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

9.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

9.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language


```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

9.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

9.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

9.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

9.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

9.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

9.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

9.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

10 Introduction to Python

10.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

10.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

10.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

10.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

10.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```


14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

10.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

10.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

10.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

10.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

10.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

11 Introduction to Python

11.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

11.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

11.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

11.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

11.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

11.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

11.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

11.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

11.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```


11.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

12 Introduction to Python

12.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

12.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

12.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

12.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

12.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

12.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

12.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

12.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

12.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

12.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

13 Introduction to Python

13.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

13.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

13.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language


```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

13.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

13.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

13.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

13.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

13.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

13.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

13.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

14 Introduction to Python

14.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

14.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

14.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

14.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

14.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```


14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

14.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

14.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

14.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

14.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```

14.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

15 Introduction to Python

15.1 Prerequisites

- There are three things we need to run the code in this lecture:
 - Python 3.10 (<https://www.python.org/downloads/>)
 - PyCharm community or professional (<https://www.jetbrains.com/pycharm/>)
 - A handful of other packages

15.2 Variables

- Let's start using a variable
- Variable names can contain only letters, numbers and underscores
- Space are not allowed in variable name
- Variables name should be short and descriptive

```
message = "Hello world!"  
print(message)
```

Hello world!

15.3 Strings

- A *string* is a series of characters
- In python, anything inside a quotes is considered a string
- We can use single or double quotes around the strings

```
message = "Python is a programming language"  
print(message)
```

Python is a programming language

```
message = 'I told my friend, "Python is a programming language"'
print(message)
```

I told my friend, “Python is a programming language”

15.3.1 Change case of string

- A lower case string can be changed to a title

```
name = "moinul islam"
print(name.title())
```

Moinul Islam

- We can also change the string to all upper or all lower

```
name = "moinul islam"
print(name.upper())
```

MOINUL ISLAM

15.4 Numbers

- Numbers are used quite often in python
 - Integers: We can add (+), multiply (*), exponent (**), and divide (/) integers in Python

```
print(2+3)
```

5

```
print(3-2)
```

1

```
print(2*3)
```

6

```
print(3**3)
```

27

```
print(3/2)
```

1.5

- Python supports the order of operation too.

```
print(2 + 3*4)
```

14

```
print((2+3)*4)
```

20

- Floats: Python calls any number with a decimal point a *float*

```
print(2*0.2)
```

0.4

- Integers and floats: When we divide two numbers, even if they are integers, you will always get a float

```
print(4/2)
```

2.0

- Underscore in numbers

- When you are writing long numbers, you can group digits using underscores

```
universe_age = 14_000_000_000  
print(universe_age)
```

14000000000

- Multiple assignments

- We can assign the value to more than one variables using just a single line

```
x, y, z = 12, 3, 5
print(x, y, z)
print(z)
```

12 3 5

5

- Constants

- A *constant* is a variable whose value stay the same throughout the life of a program.
- Python does not have built in command for constant
- However, a variable name with all capital letters treated as constant

```
ELON_MASK = 1_000_000_000_000
print(ELON_MASK)
```

1000000000000

15.5 Comments

- In Python, hash (#) indicates a comment.
- Anything following a # mark in your code is ignored by Python

```
# Say hello to your friends
print("Hello friends")
```

Hello friends

15.6 Lists

- List the elements of a variable

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

- Access an element from a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0])
```

```
trek
```

15.6.1 Neat outcome

- You can format the element “trek” even more neatly by using title() method

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[0].title())
```

```
Trek
```

15.6.2 Print string

- You can print the strings that you are interested
- It starts counting from zero

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[1])
```

```
cannondale
```

- Python has special syntax to call the last item of a list

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
print(bicycles[-1])
```

```
specialized
```


15.7 Functions

```
bicycles = ["trek", "cannondale", "redline", "specialized"]  
message = f"my first bicycle was a {bicycles[0].title()}"  
print(message)
```

my first bicycle was a Trek

- Modifying elements in a list

References