Python Basics

Computing for Data Analytics (CPSC 4800)

Mourad Bouguerra
mourad.bouguerra@me.com

Langara College

Summer 2023

Lesson's Outline

1. **Lesson's Learning Objectives**
2. **Introduction**
   - Python History
   - Python Features
   - Read Evaluate Print Loop (REPL)
   - Integrated Development Environment (IDE)
3. **Python Basics**
   - Hello World Program
   - Identifiers
   - Naming Conventions
   - Data Types
   - Python Script
     - Python Comments
     - Statements
     - Expressions
     - Operators

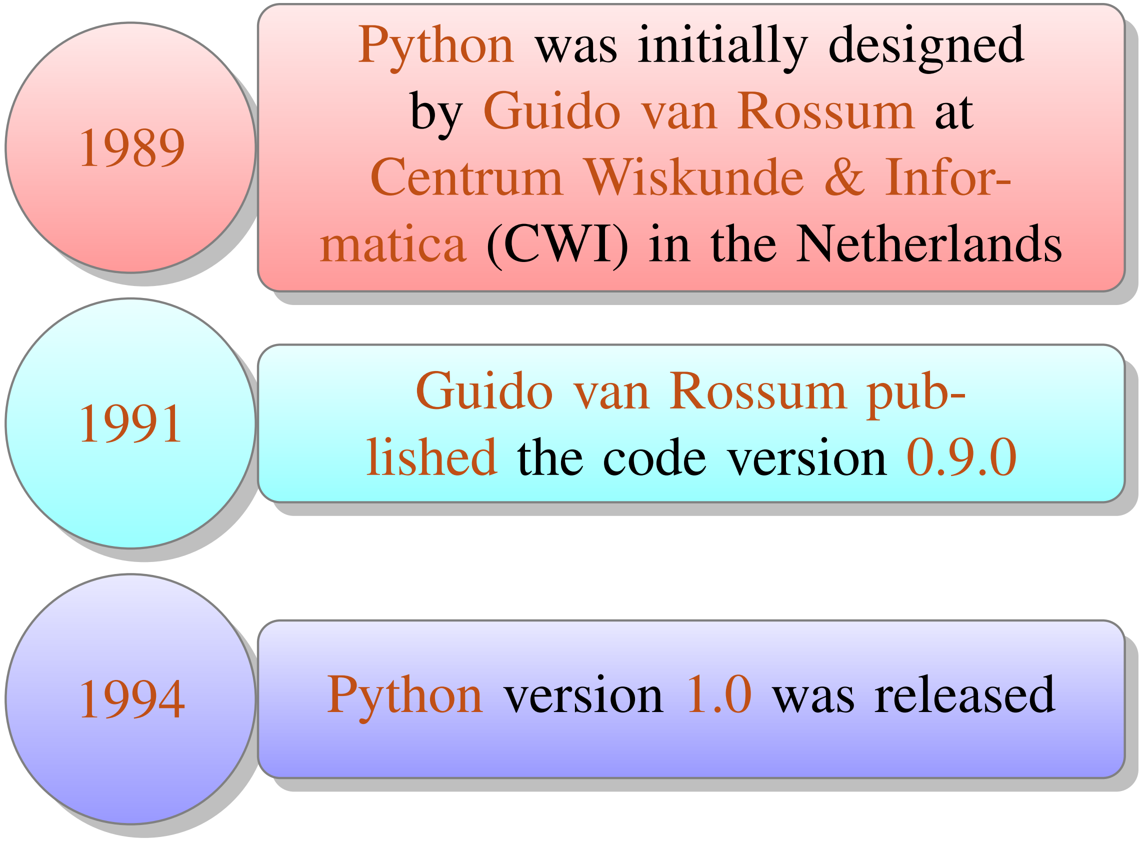
Learning Objectives

Learning Objectives

✎ Upon completion of this lesson, you will learn:

- ❐ Python identifiers and naming conventions
- ❐ Python builtin data types
- ❐ Python different comments
- ❐ Python expressions and statements
- ❐ Python operations

Python History



1989

Python was initially designed by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands

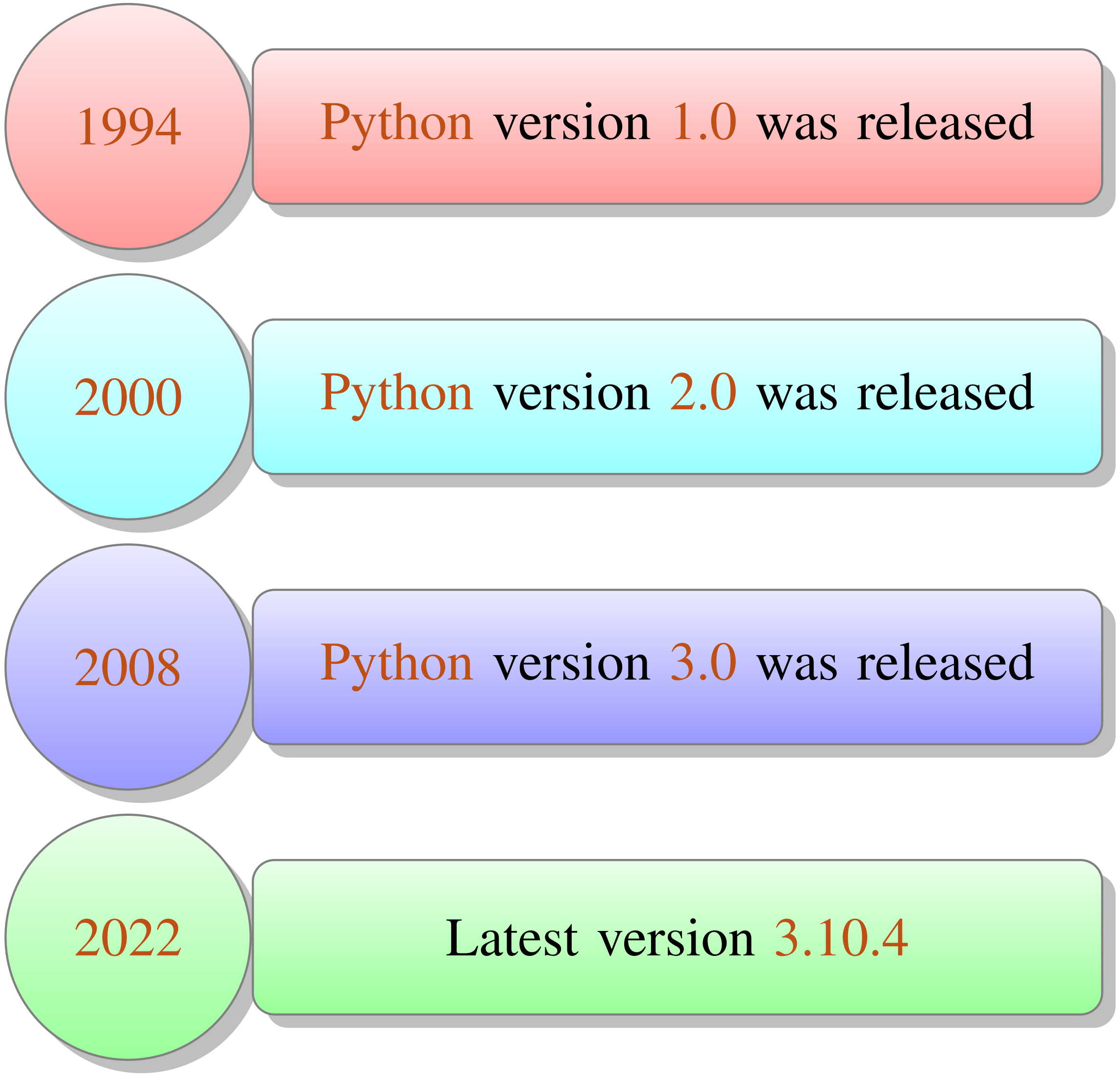
1991

Guido van Rossum published the code version 0.9.0

1994

Python version 1.0 was released

Python History



1994 Python version 1.0 was released

2000 Python version 2.0 was released

2008 Python version 3.0 was released

2022 Latest version 3.10.4

Python Features

Python Features

✎ Python is a multi-paridigm language

- ❐ Object-Oriented
- ❐ Structured
- ❐ Functional

➦ Python provides simple syntax[a]

➦ Python is highly extensible through modules

➦ Python as interpreted language is easy to test code[b]

[a] Simple is Better than Complex Python 's design principle.
[b] Python does not require the compilation step.

Python Features

Python Features

❒ Python has builtin support for

- ➦ Data Science
- ➦ Security
- ➦ Web Programming
- ➦ Databases

REPL Environment

Read Evaluate Print Loop (REPL)

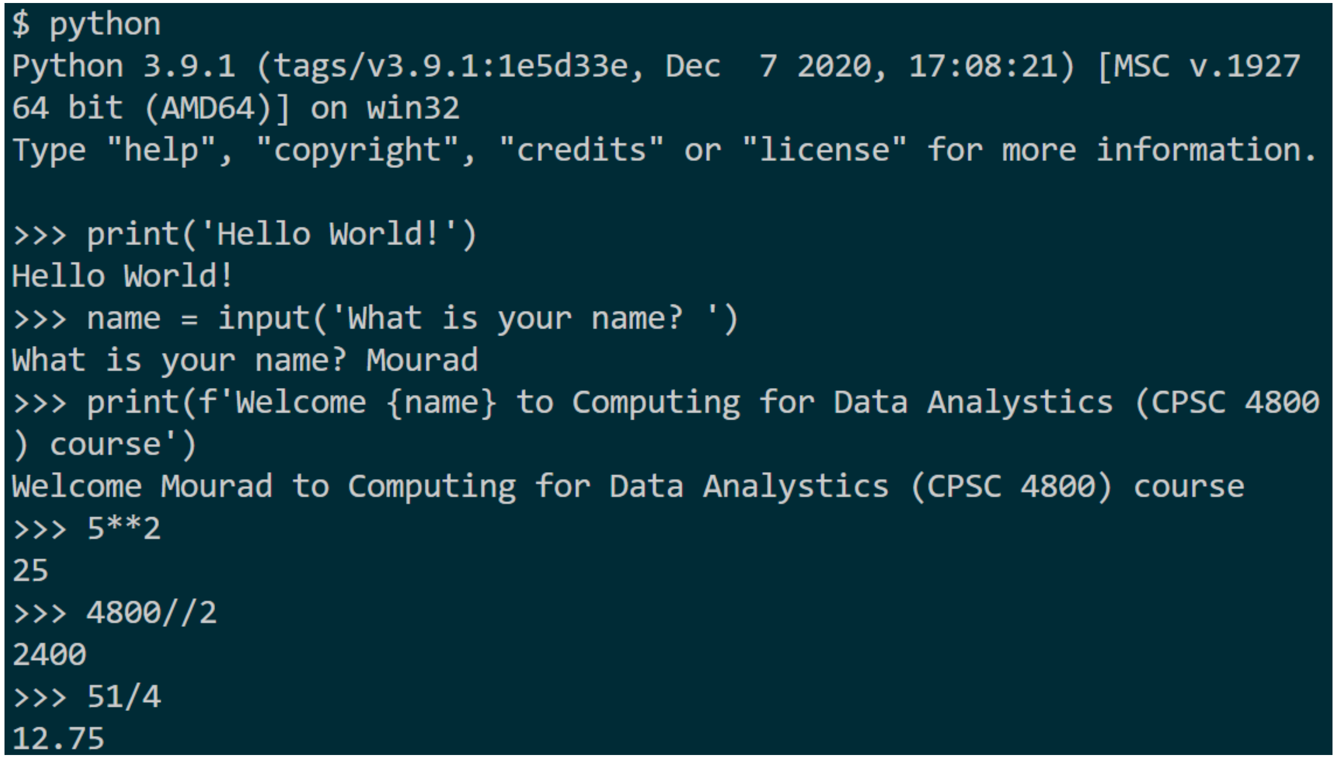
- ❒ An interactive interpreter that allows fast experimentation with a programming language
  - ➦ interpreted (scripting) languages
    - ✔ `Python, JavaScript, Ruby, Perl, .........`
- ❒ Unlike compiled languages
  - ➦ `C, C++, Java, C#, .........`
    - ✔ `Require compilation step`

REPL Environment

REPL Environment

- ❒ Python provides two REPL environments
  - ➦ Interactive shell
  - ➦ Jupyter notebook
- ❒ Interactive shell is a command-line interface (CLI)
- ❒ Jupyter notebook is a graphical user interface (GUI)

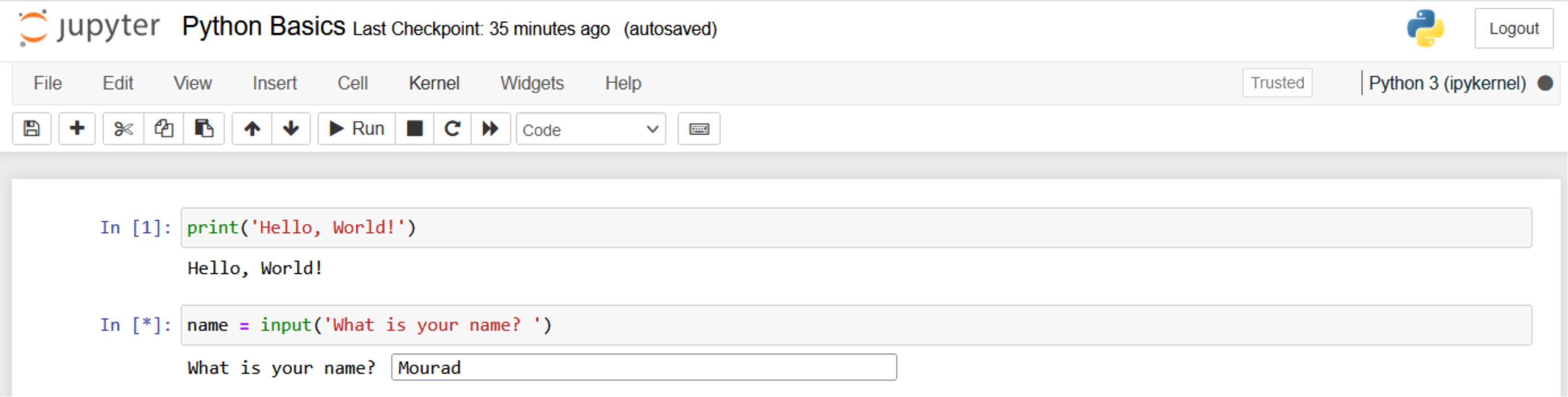
CLI REPL



```
$ python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> print('Hello World!')
Hello World!
>>> name = input('What is your name? ')
What is your name? Mourad
>>> print(f'Welcome {name} to Computing for Data Analystics (CPSC 4800
) course')
Welcome Mourad to Computing for Data Analystics (CPSC 4800) course
>>> 5**2
25
>>> 4800//2
2400
>>> 51/4
12.75
```

GUI REPL



```
In [1]: print('Hello, World!')

        Hello, World!

In [*]: name = input('What is your name? ')

        What is your name? Mourad
```

Integrated Development Environment (IDE)

IDE

❒ An Integrated Development Environment (IDE)

- ➦ a software that assists programmers in
  - ✔ developing, running & testing programs

❒ An Integrated Development Environment (IDE) includes

- ➦ Text Editor
- ➦ Debugger
- ➦ Automation Tools
- ➦ IntelliSense

Python IDEs

Microsoft Visual Studio

- Visual Studio IDE

Spyder

- Spyder IDE

Python IDEs

Microsoft Visual Studio

- ❐ Visual Studio IDE

PyCharm

- ❐ PyCharm IDE

Integrated Development Environment (IDE)

IDE

- Type the following in an REPL environment

```
print('Hello, World!')
```

- Run the line/cell[a] from the REPL environment
- Use an IDE's text editor to create a source file `hello_world.py`
- Run the source file

```
python hello_world
```

[a] Press enter in the shell, and click the run button in the Jupyter notebook.

Hello World Program

- Use a text editor to create a source file `hello_world.py`

```
print(’Hello, World!’)
```

Python Keywords

Keywords

- ❐ Python keywords are reserved words
- ❐ Python keywords have specific meaning in the Python language
  - ➦ Cannot be used by programmer
- ❐ Python is case-sensitive language

Python Keywords

Keywords

- Python has 33 keywords

```
1  and
2  as
3  assert
4  break
5  class
6  continue
7  def
8  del
9  elif
10 else
11 except
```

```
12 FALSE
13 finally
14 for
15 from
16 global
17 if
18 import
19 in
20 is
21 lambda
22 None
```

```
23 nonlocal
24 not
25 or
26 pass
27 raise
28 return
29 TRUE
30 try
31 while
32 with
33 yield
```

Python Identifiers

Identifiers

- ❒ An identifier is a programmer-defined name
  - ➦ used in the Python code/script
- ❒ A valid Python identifier is a series of characters consisting of
  - ➦ letters a to z and A to Z
  - ➦ digits 0, 1 to 9
  - ➦ underscore _
    - ✔ CANNOT be a Python keyword
    - ✔ CANNOT start with a digit
    - ✔ CAN have any length

Class Activity

✎ Identify valid Python identifiers

① 4You
② *_NumberOfBytes*
③ *xyz999*
④ *private*
⑤ *firstName*
⑥ *room location*
⑦ *Xy49Ztyew*
⑧ *class*
⑨ *SimpleEncryption*
⑩ *PI*

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Naming Conventions

Identifiers

- ❒ An identifier should be descriptive and readable
- ❒ An identifier should comply with the Python style guide
  - ➦ Python Enhancement Proposal (PEP 8)[a]

[a] `https://peps.python.org/pep-0008/`

Naming Convention	Example
lowercase	`computingfordataanalytics`
UpperCamelCase	`ComputingForDataAnalytics`
lowerCamelCase	`computingForDataAnalytics`
UPPERCASE_WITH_UNDER_SCORE	`COMPUTING_FOR_DATA_ANALYTICS`
lowercase_with_under_score	`computing_for_data_analytics`

Python Enhancement Proposal (PEP 8)

Module

❒ A Python module[a] is

➦ a Python source file

[a] It is called a package in other programming languages.

Identifier	Naming Convention	Example
Module	lowercase_with_underscores	`hello_world.py`

Class Activity

✎ Which of the following name conventions is more readable for combined words

① lowercase
② lowerCamelCase
③ UpperCamelCase
④ Upper_Case

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

✎ Which name convention is used for each of these identifiers?

① NumberOfBytes
② MAXIMUM_CAPACITY
③ computeFiveNumberStatistics()
④ SimpleEncryption
⑤ get_total()
⑥ NumberOfRequests

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Python Data Types

Data Types

❐ Python has two ways of storing data

① variable

➦ refers to a memory location that has a value that can change during program execution

```
pep_8_url = 'https://peps.python.org/pep-0008/'
course_credit =
is_transferable = True
```

② constant

➦ refers to a memory location that has a value that does NOT change during program execution

```
PI =
EULER_NUMBER =
```

Python Data Types

Constant

- ❒ Constants are usually declared in a module
  - ➦ a separate source file and then imported

```python
# declared in source file constant.py
PI =
EULER_NUMBER =
```

- ❒ Constants are imported to another module
  - ➦ by using the import keyword

Python Enhancement Proposal (PEP 8)

Identifier	Naming Convention	Example
Module	lowercase_with_underscores	`hello_world.py`
Variable	lowercase_with_underscores	`pep_8_url`
Constant	UPPERCASE_WITH_UNDER_SCORE	`EULER_NUMBER`

Python Built-in Data Types

Category Type	Identifiers	Example
Text/String	str	pep_8_url = 'https://peps.python.org/pep-0008/
Numeric	int, float, complex	course_credit =
Boolean	bool	is_transferable = True
Sequence	list, tuple, range	vowels= 'a' 'e' 'i' 'o' 'u'
Mapping	dict	course= 'code' 'CPSC 4800' 'credit'
Set	set	my_set= 'CPSC 4800' True

Python Built-in Data Types

Category Type	Identifiers	Example
Text/String	`str`	`pep_8_url = str 'https://peps.python.org/pep-0`
Numeric	`int, float, complex`	`course_credit = int`
Boolean	`bool`	`is_transferable = bool True`
Sequence	`list,tuple,range`	`vowels=list 'a' 'e' 'i' 'o' 'u'`
Mapping	`dict`	`course=dict code='CPSC 4800' credit=`
Set	`set`	`my_set=set    'CPSC 4800'  True`

Python Data Types

Dynamically Type

- ❒ Python is dynamically type language
  - ➦ Does not have to declare the type of the variable explicitly
  - ➦ Data type can be inferred from the declaration
- ❒ To check the data type in Python:
  - ➦ `type()`

```
course= 'title' 'Computing for Data Analytics' 'code' 'CPSC 4800' 'credit'
type course
#output -> <class 'dict'>
```

Python Data Types

Dynamically Type

- ❒ Python type is attached to the value of the variable
  - ➦ The value of the variable can change its type
  - ➦ The variable is a container

Homework

- ❐ To compute the minimum size in bytes for int data type, you use the following code

```
import sys
int_size = sys getsizeof int
```

- ❐ Complete the following Python program to compute the minimum size of all Python data types

```
import sys
int_size = sys getsizeof int
float_size = sys getsizeof float
print f'Minimum Size of Data Types'
print f'=========================='
print f'Minimum size of int type is {int_size} bytes'
print f'Minimum size of float type is {float_size} bytes'
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Python Data Types

Python Script/Program

❒ Python source file `.py`[a] consists of

① comments

✔ used to document your script

② statements

✔ instructions to the computer to carry out some tasks

[a] Jupyter notebook has the extension .ipynb.

Python Script

Python Comments

- ❐ It is a good programming practice to use comments in your program[a]
- ❐ Three types of Python comments
  - ➦ in-line comments
  - ➦ block comments
  - ➦ documentation string comments
- ❐ Comments should be complete sentences that explain the logic of the code

[a] Follow PEP 8 for Python Style Guide.

Python Script

In-Line Comment

- An In-line comment is single-line comment
  - can be at the same line as the statement or by itself
- in-line comment should be preceded by the hash symbol

```python
 import pandas # This is an line comment
```

Python Script

Block Comment

- A block comment is multiline comment that
  - spreads over two or more lines
- Eachline is preceded by the hash symbol

```
# A block comment first line
# A block comment second line
# A block comment third line
```

Python Script

Documentation String Comment

❒ A documentation string comment is a multiline comment that

- ➡ spans over two or more lines

❒ A documentation string comment should be enclosed by

- ➡ three single quotes, i.e., ’’’

```
'''
This is a documentation string comment
'''
```

Python Statements

Python Statement

- ❒ Python does not requires statement termination printable character
  - ➦ such as semi-colon (;)
- ❒ Python statement end with the end of the line
- ❒ To split a statement over multiple lines
  - ➦ backslash (\) can be used

```
welcome = 'Welcome to Computing for Data Analytics (CPSC 4800)\n\
at Langara College\n\
summer 2022'
print welcome
```

Python Statements

Python Statement

- Semi-colon (;) can be used to separate
  - multiple statements on the same line

```
x=5;y=10;print(f’(x,y)=({x},{y})’)
```

- This should be avoided
  - to make your code readable

Python Statements

Python Block Statement

- Block or compound statement refers to
  - two or more statements that is grouped together

```python
for x in range(100000):
    if x == 100:
        print(f'x={x}')
        break
    print(f'x={x}')
```

- Python uses identation to identify block statement

Python Expressions

Python Expressions

- An expression is anything that evaluates to a value
- Simple expression consists of a single item
- Complex expressions are constructed from simple expressions using operators

```
# The string ’Python Expression’ is a simple
   expression
# evaluates to iteself
title = ’Python Expression’
MAXIMUM = 5
rate = .05
# The following complex expression
x = 100 + MAXIMUM*rate
```

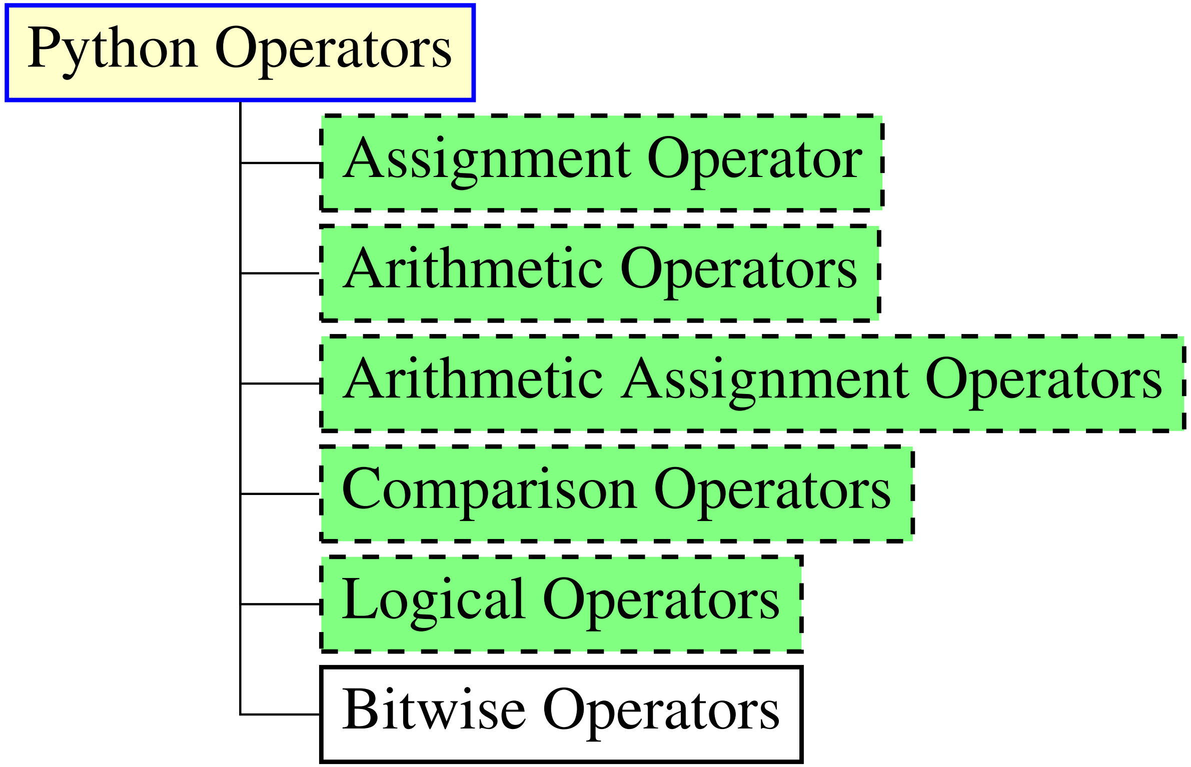
Python Operators

Python Operators

- ❒ An operator is a symbol that instructs Python to perform some operation on one or more operands
- ❒ An operand can be any valid expression
- ❒ Three types of operators
  - ➦ Unary opearator: one operand
  - ➦ Binary opearator: two operands
  - ➦ Ternary opearator: three operands

```python
x = 5 # (=) binary operator
y = -x # (-) unary operator
z = y ** x # (**) binary operator
print(f'(x,y,z)=({x},{y},{z})')
```

Python Operators



Python Operators

Assignment Operator

Arithmetic Operators

Arithmetic Assignment Operators

Comparison Operators

Logical Operators

Bitwise Operators

Python Assignment Operator

Python Assignment Operator

- Assignment operator denoted by $=$ operands
- Assign the value of the Right Hand Side (RHS) to Left Hand Side (LHS)

  ➦ $LHS = RHS$
- The assignment expression is evaluated to the value of LHS
- The RHS value does NOT change

```python
x = y = z = 2**8
print(f'(x,y,z)=({x},{y},{z})')
```

Python Arithmetic Operators

Python Binary Arithmetic Operators

Operator	Symbol	Description	Example
Addition	+	Adds its two operands	`a +b`
Subtraction	−	Subtracts the second operand from the first operand	`a -b`
Multiplication	∗	Multiplies its two operands	`a *b`
Division	/	Divides the first operand by the second operand	`a/b`
Floor (Integer) Division	//	Divides the first operand by the second operand returns the integer part	`a//b`
Power	∗∗	The first operand to the power of the second operand	`a**b`
Modulus	%	Gives the remainder when the first operand is divided by the second operand	`a % b`

Python Arithmetic Operators

Python Arithmetic Asignment Operators

Shorthand	for
a+=b	a=a+b
a-=b	a=a-b
a*=b	a=a*b
a/=b	a=a/b
a//=b	a=a//b
a**=b	a=a**b
a%=b	a=a% b

Python Comparison Operators

Python Comparison Operators

- Comparison or Relational operators are used to compare expressions
  - `Is x greater than 100?`
  - `Is x equal to y?`
- An expression containing a comparison operator evaluates to
  - ✔ `True` or `False`

```
x=1000
y=1e3
print(f'x==y is evaluated to {x==y}')
print(f'type(x)==type(y) is evaluated to {type(x)==type(y)}')
```

Python Comparison Operators

Python Comparison Operators

Operator	Description	Example
$==$	Is first operand equal to second operand?	$a == b$
$>$	Is first operand greater than second operand?	$a > b$
$<$	Is first operand less than second operand?	$a < b$
$\geq$	Is first operand greater than or equal second operand?	$a \geq b$
$\leq$	Is first operand less than or equal second operand?	$a \leq b$
$! =$	Is first operand not equal to second operand?	$a! = b$

Python Logical Operators

Python Comparison Operators

- ❒ Logical operators allows to
  - ➦ combine one or more boolean expressions into a single boolean expression
- ❒ Three logical operators
  - ➦ The conjunction Logical AND denoted by `and`
  - ➦ The disjunction Logical OR denoted by `or`
  - ➦ The Negation Logical NOT denoted by `not`

```
x = True
y = False
print f'x and y is evaluated to {x and y}.'
print f'x or y is evaluated to {x or y}.'
print f'not x is evaluated to {not x}.'
print f'not y is evaluated to {not y}.'
```

Python Logical Operators

Python Logical Operators

Operator	Symbol	Example	Description
Logical AND	`and`	`exp1 and exp2`	the result is true if and only if both operands are true
Logical OR	`or`	`exp1 or exp2`	the result is true if and only one the operands is true
Logical NOT	not	`not exp`	the result is true if the operand is false,otherwise is false

Python Logical Operators

Python Operator Precedence

- Operator precedence defines rules that specify
  - ➦ the order the operations are performed
- Each operator has a specific precedence
- Operator with thhighest precedence is performed first

Arithmetic Operator Precedence		
Operator	Precedence Order	Associativity
**	1	Left to right
+, −	2	Left to right
*, /, //, %	3	Left to right
+, −	4	Left to right

Python Operator Precedence

- ❒ Operator precedence defines rules that specify
  - ➦ the order the operations are performed
- ❒ Each operator has a specific precedence
- ❒ Operator with thhighest precedence is performed first

Comparison Operator Precedence		
Operator	Precedence Order	Associativity
<, <=, >, >=	1	Left to right
==,!=	2	Left to right

Python Logical Operators

Python Operator Precedence

- ❐ Operator precedence defines rules that specify
  - ➦ the order the operations are performed
- ❐ Each operator has a specific precedence
- ❐ Operator with thhighest precedence is performed first

Logical Operator Precedence		
Operator	Precedence Order	Associativity
`not`	1	`Left to right`
`and`	2	`Left to right`
`or`	3	`Left to right`

Python Logical Operators

Python Operator Precedence

Highest to Lowest Precedence	
Operator	Description
()	Grouping
**	Power
−, +	Unary Positive, Negative
*, /, //, %	Multiplication, division, and reminader
+, −	Addition, subtraction
==, =,>,<,>=,<=!	Comparison
not	Logical NOT
and	Logical AND
or	Logical OR

Class Activity

- Given the following Python variables

```python
x = 13
y = 5
```

- evaluate the following Python expressions

```python
w = 13**2/5
z = 13//y
x //=3
y *=w
a = isinstance(w,int)
b = isinstance(w,float)
c= a and b
d = a or b
e = not (w > (y*2) ) and (y > w)
f = (100 == 1e2) or (2**8 >= 2e2)
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand