PRIME INTUIT

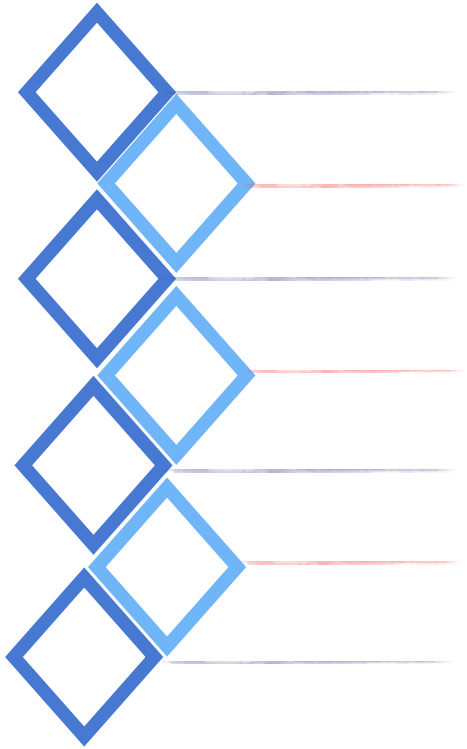
Finishing School

Variable Types and Basic Operations

Python – Session – 2

PRIME INTUIT
Finishing School

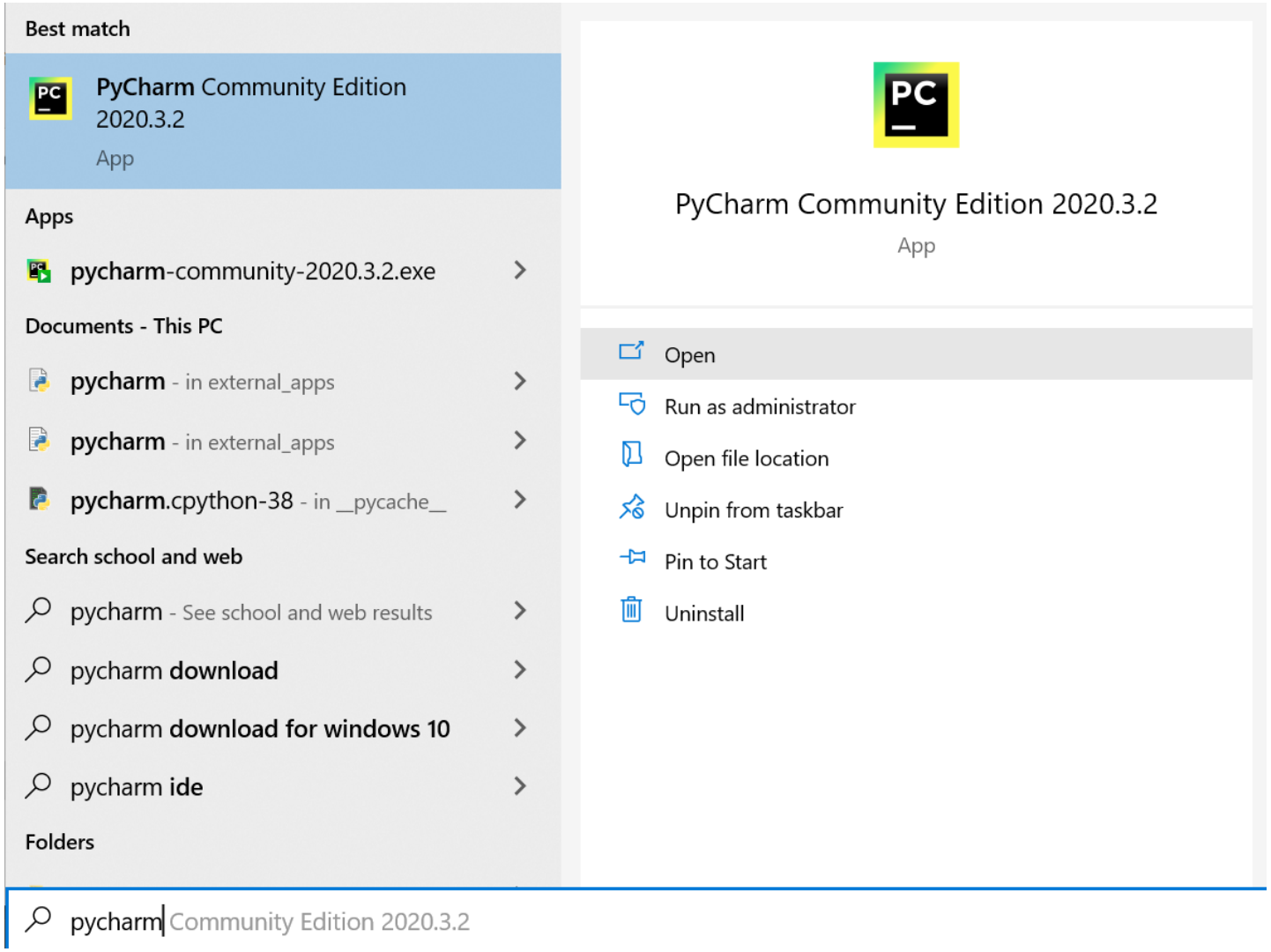
Agenda



- Setting Appearance
- Setting Up Path
- Creating a Project
- Lines and indentation
- Basic data Types
- Basic Operations
- Data Type conversion

PRIME INTUIT
Finishing School

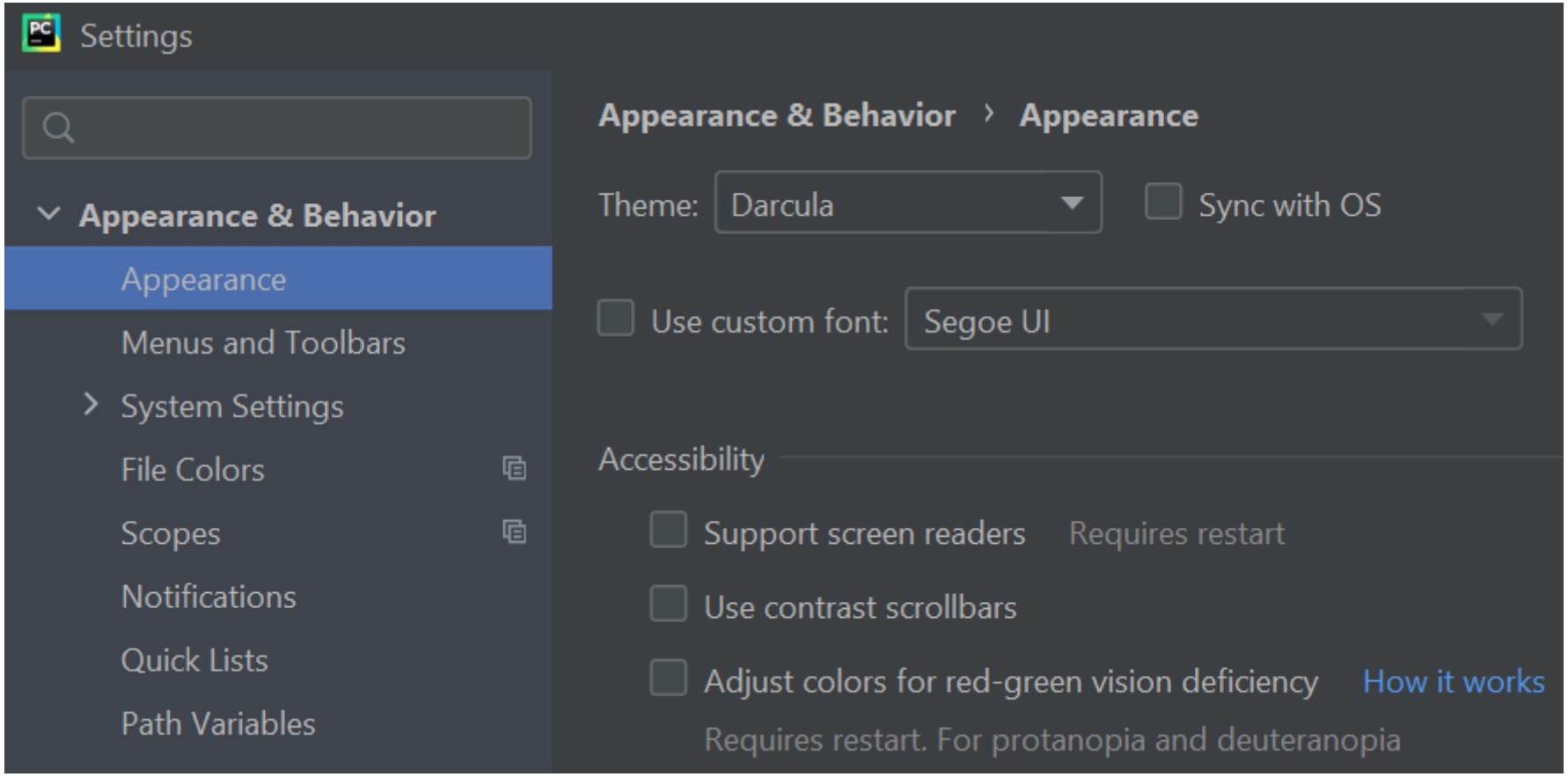
Working with Pycharm



PRIME INTUIT
Finishing School

Check this ?

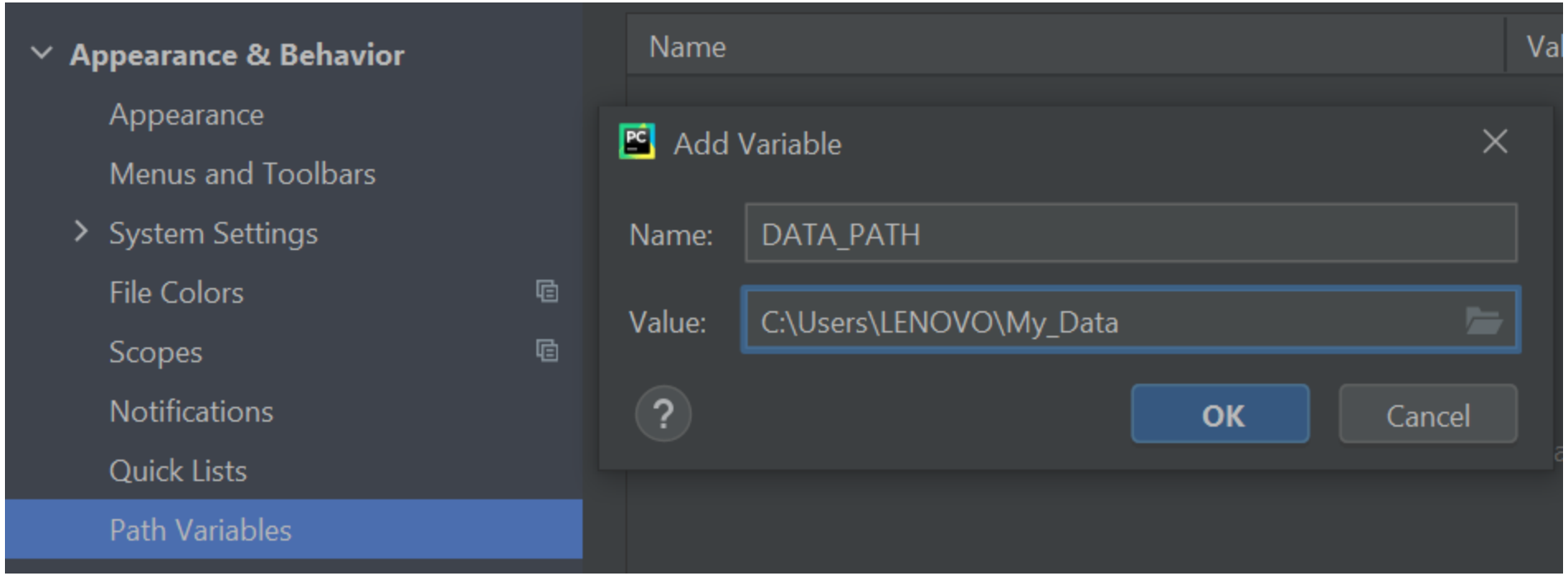
Setting Appearance



PRIME INTUIT
Finishing School

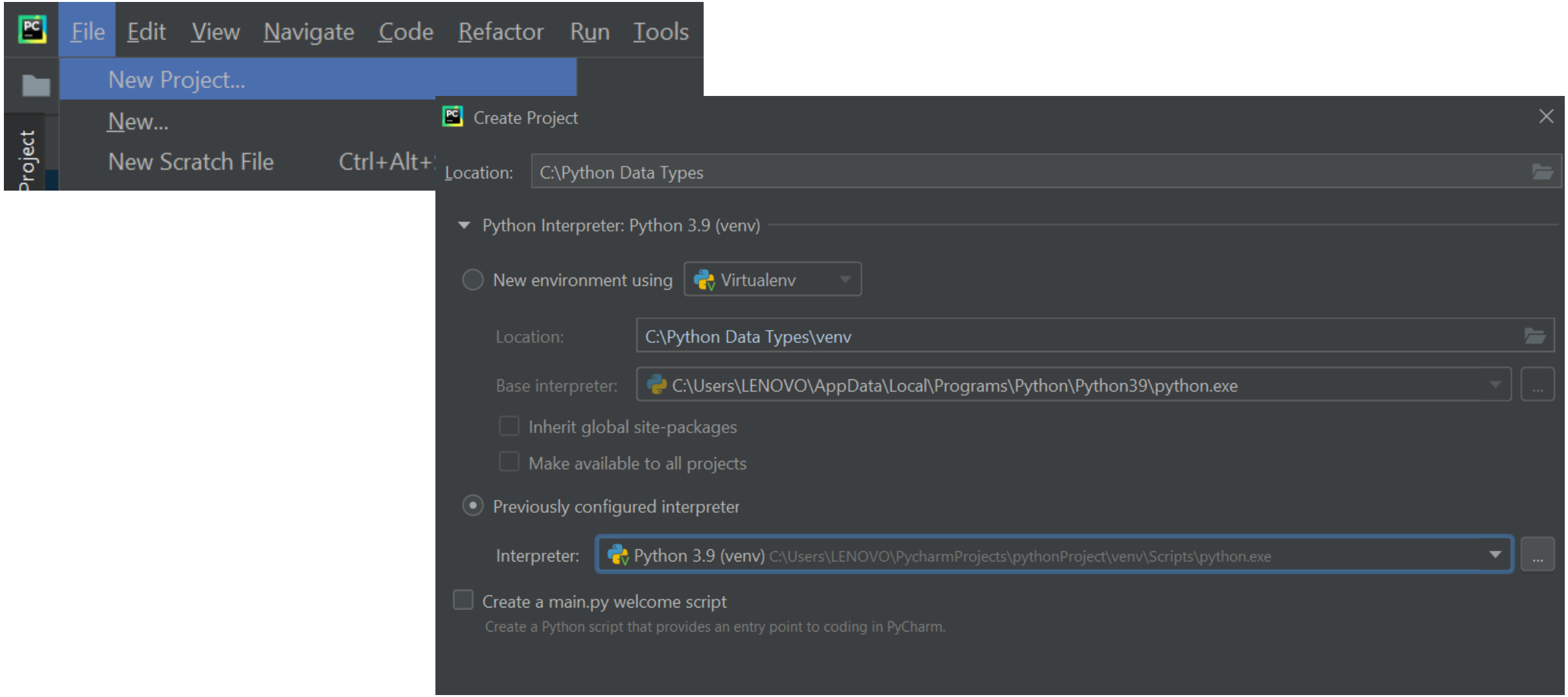
Check this

Setting up Path



PRIME INTUIT
Finishing School

Creating a Project



PRIME INTUIT
Finishing School

Python – Reserved Words

Reserved words are words used by Python Language itself & Hence, you cannot use them as constants or variables or any other identifier names. All Python keywords contain lowercase letters only.

Reserved words are also referred to as Python vocabulary

PRIME INTUIT
Finishing School

Python Vocabulary / Reserved Words

```
and       del       global    not       with
as        elif      if        or        yield
assert    else      import    pass
break     except    in        raise
class     finally   is        return
continue  for       lambda    try
def       from      nonlocal  while
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, **PrimeIntuit** and **primeintuit** are two different identifiers in Python.

Python Identifiers

Naming Conventions for Python identifiers:

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is private.
- Starting an identifier with two leading underscores indicates a strong private identifier.
- If the identifier also ends with two trailing underscores, the identifier is a languagedefined - special name.

More Relevant for OOP

Python – Lines and Indentation

Python does not use braces({}) to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The nuamber of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example

```
If True:
      print ("Answer")
      print ("True")
else:
      print "(Answer")
      print ("False")
```

```
if True:
    print ("Answer")
    print ("True")
else:
    print "(Answer")
  print ("False")
```

Thus, in Python all the continuous lines indented with the same number of spaces would form a block.

PRIME INTUIT
Finishing School

Python Vocabulary / Reserved Words

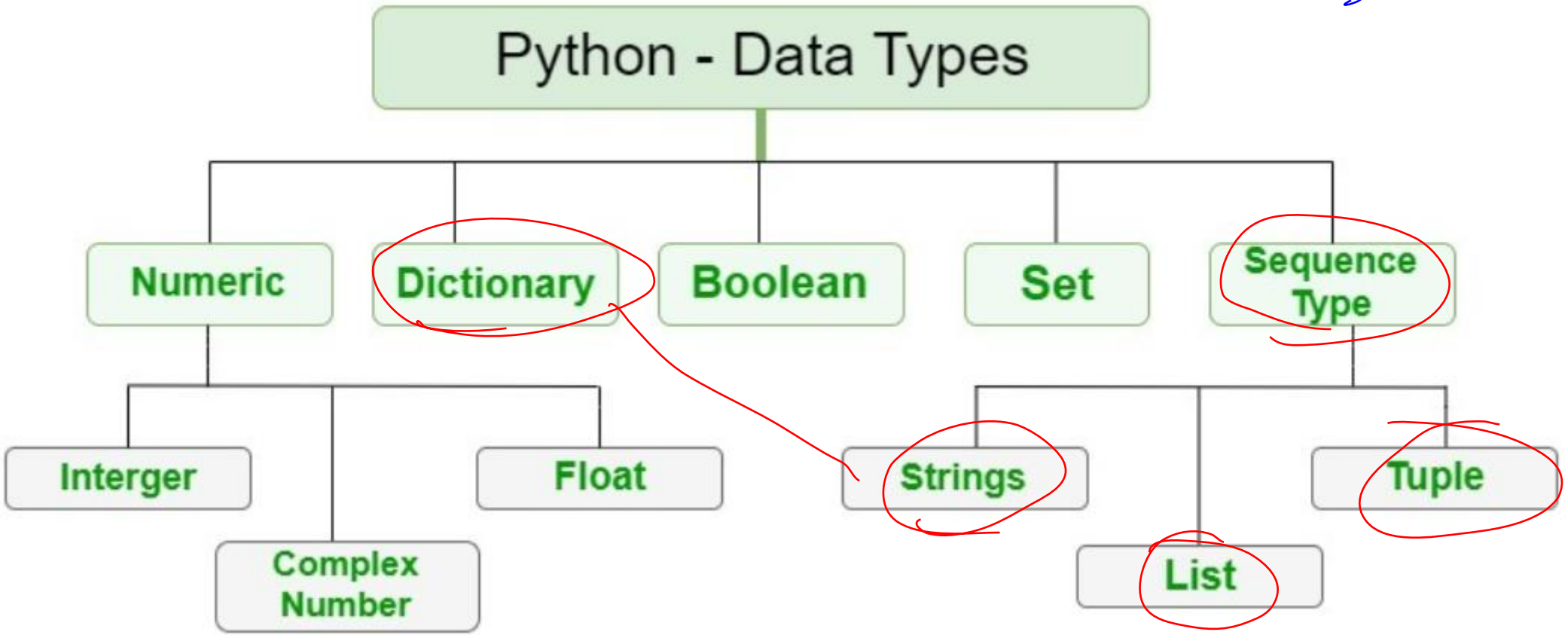
```
and       del       global    not       with
as        elif      if        or        yield
assert    else      import    pass
break     except    in        raise
class     finally   is        return
continue  for       lambda    try
def       from      nonlocal  while
```

PRIME INTUIT
Finishing School

Python Data Types

Following are the standard or built-in data type of Python:

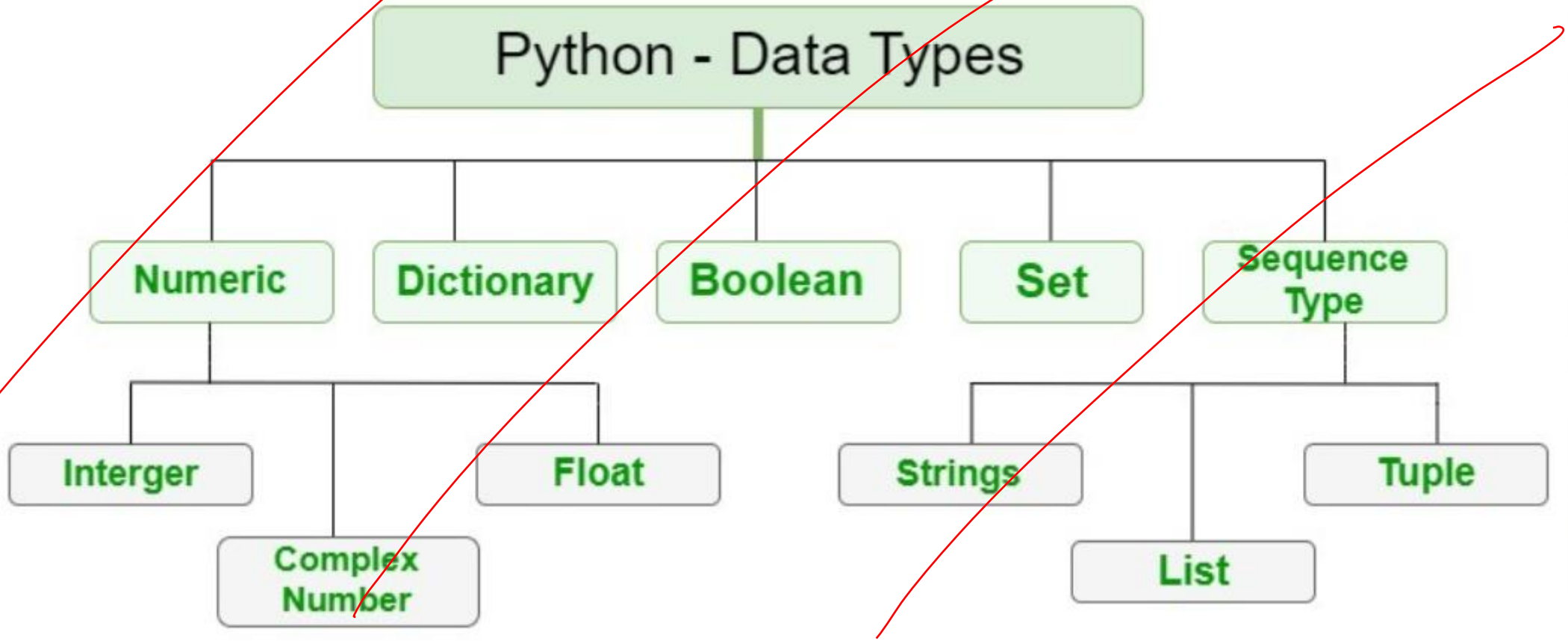
Dictionary is not seq type



PRIME INTUIT
Finishing School

Python Data Types

Following are the standard or built-in data type of Python:



PRIME INTUIT
Finishing School

Variables

Variables are nothing but reserved memory locations to store values. It means that when you create a variable, you reserve some space in the memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the **reserved memory**. Therefore, by assigning different data types to the variables, you can store integers, decimals or characters in these variables.

Python variables do not need **explicit declaration** to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

PRIME INTUIT
Finishing School

Variables

Programmers generally choose names for their variables that are meaningful they document what the variable is used for.

```
Message = 'And now for something completely different'
n = 17
pi = 3.1415926535897932
```

Variable names can be as long as you like. They can contain both letters and numbers, but they ~~can't begin with a number.~~ It is legal to use uppercase letters, but it is conventional to use only lower case for variables names. The underscore character, _, can appear in a name.

PRIME INTUIT
Finishing School

Assigning Values to Variables

Check Pycharm programs & DPPs & add

“There was a small boy named **Soma**,
He used to sell **Flowers** after his School,
every day after **4** PM, He used go to market on his Bicycle
and sell **Flowers** until it turned dark.
Today he made a profit of Rs **200.50** ”

```
print("There was a small boy named Soma, ")
print("He used to sell Flowers after his School, ")
print("every day after 4 PM, He used go to market on his Bicycle ")
print("and sell Flowers until it turned dark. ")
print("Today he made a profit of Rs 200.50 ")
```

Assigning Values to Variables

```
name = "suresh"
product = "Samosa"
time = 5
profit = 180.25

print("There was a small boy named", name)
print("He used to sell", product, "after his School, ")
print("every day after",time, ", He used go to market on his Bicycle ")
print("and sell", product," until it turned dark. ")
print("Today he made a profit of Rs: ", profit)
```

PRIME INTUIT
Finishing School

Assigning Values to Variables

```
name = "suresh"
product = "Samosa"
time = 5
profit = 180.25

print("There was a small boy named", name)
print("He used to sell", product, "after his School, ")
print("every day after",time, ", He used go to market on his Bicycle ")
print("and sell", product," until it turned dark. ")
print("Today he made a profit of Rs: ", profit)
```

PRIME INTUIT
Finishing School

Expressions and statements

An **expression** is a combination of values, variables, and operators. A value all by itself is considered an expression, and so is a variable, so the following are all legal expressions:

```
>>> 42
42
>>> n
17
>>> n + 25
42
```

When you type an expression at the prompt, the interpreter evaluates it, which means that it finds the value of the expression. In this example, n has the value 17 and n + 25 has the value 42.

Expressions and statements

A **statement** is a unit of code that has an effect, like creating a variable or displaying a value.

```
>>> n = 17
>>> print(n)
```

The first line is an assignment statement that gives a value to n. The second line is a print statement that displays the value of n. When you type a statement, the interpreter executes it, which means that it does whatever the statement says. In general, statements don’t have values.

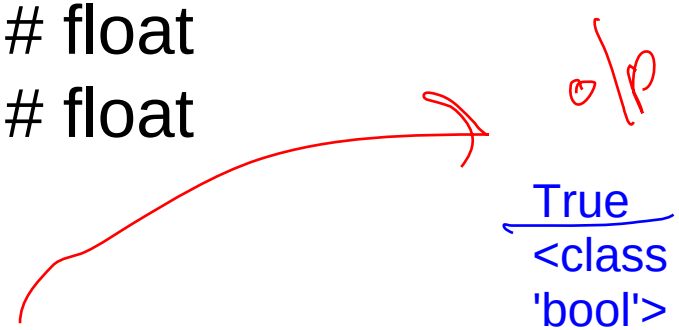
PRIME INTUIT
Finishing School

Assigning Values to Variables

```
a = 12 # int
b =14
c = a+b
print (c)

x = 1.25 # float
y = 2.75 # float

z = y>x
print(z)
print(type(z))   # type statement lets you know the type of data
```



PRIME INTUIT
Finishing School

Multiple Assignments

```
a = b = c =1
print(a,b,c)
```

```
x,y,z = "Tarun", 22, 4050.34
print(x,y,z)
```

Number data types store numeric values. Number objects are created when you assign a value to them.

```
var1 = 101
var 2 = 202
print(var1, var2)
```

You can also delete the reference to a number object by using the **del** statement
You can delete a single object or multiple objects by using the **del** statement.

Different Number Types

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	-.6545+0j
080	32.3+e18	
-0490	-90.	

A complex number consists of an ordered pair of real floating-point numbers denoted by x + yj, where x and y are real numbers and j is the imaginary unit.

Complex numbers

```
x = 3
y = 5
z = complex(x,y);
print(z)
print("Real part of complex number z is:", z.real)
print("Imaginary part of complex number z is:",z.imag)
```

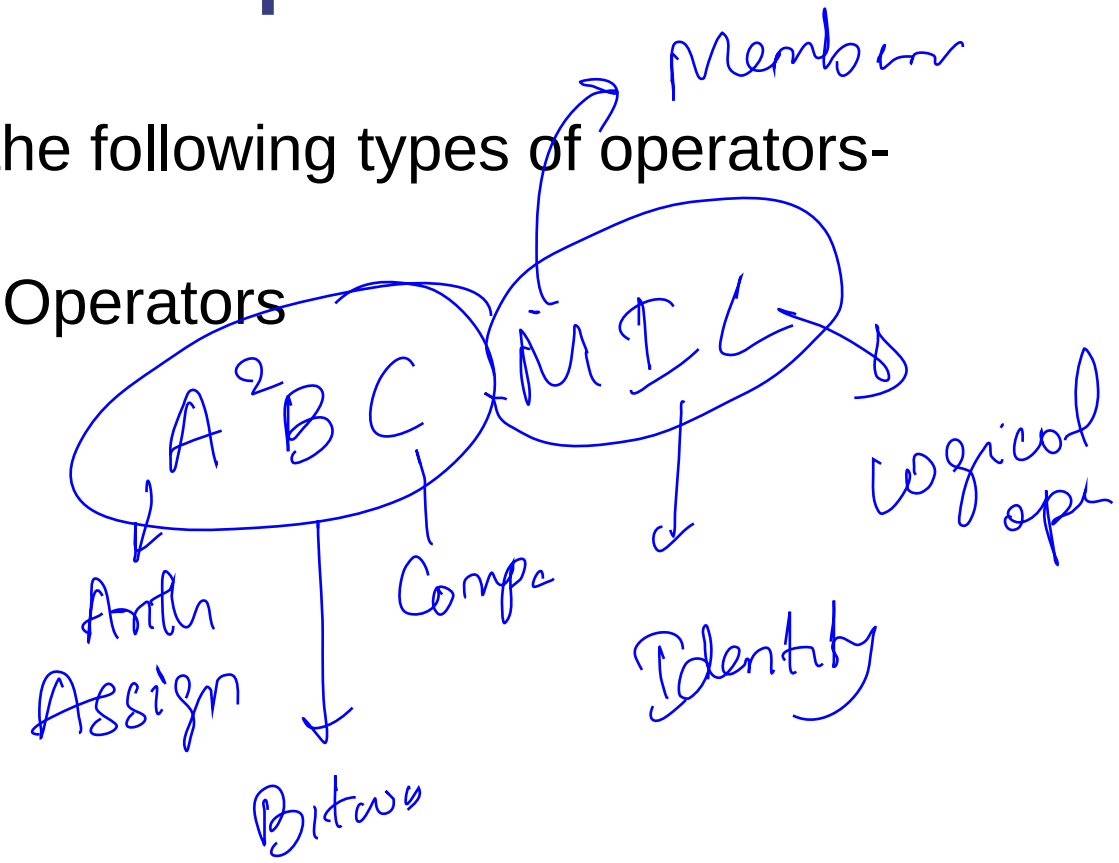
Phase of complex number

Geometrically, the phase of a complex number is the **angle between the positive real axis and the vector representing complex number**. This is also known as **argument** of complex number. Phase is returned using **phase()**, which takes complex number as argument. The range of phase lies from **-pi to +pi.** i.e from **-3.14 to +3.14**.

Python operators

Python language supports the following types of operators-

- ✓ Arithmetic Operators
- ✓ Comparison (Relational) Operators
- ✓ Assignment Operators
- ✓ Logical Operators
- ✓ Bitwise Operators
- ✓ Membership Operators
- ✓ Identity Operators



PRIME INTUIT
Finishing School

Python Arithmetic operators

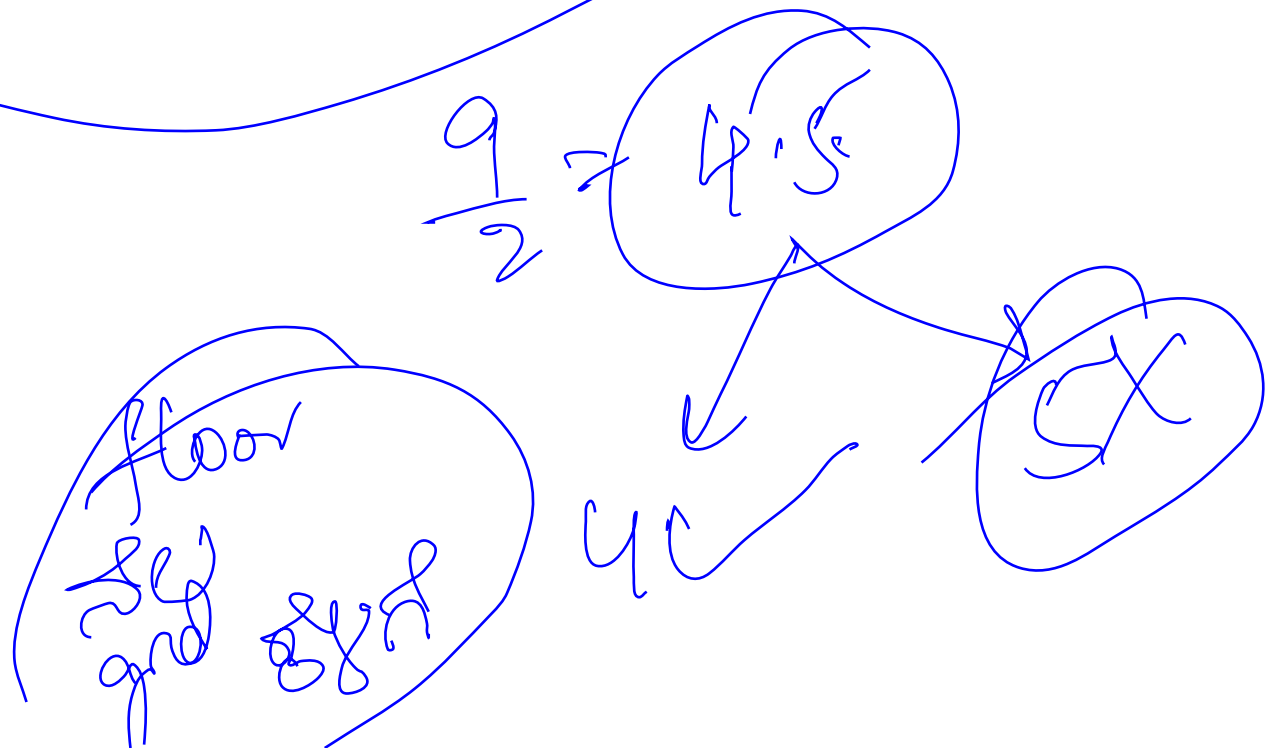
Assuming a = 10 and b = 21

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 31
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -11
* Multiplication	Multiplies values on either side of the operator	a * b = 210
/ Division	Divides left hand operand by right hand operand	b / a = 2.1
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 1
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20

PRIME INTUIT
Finishing School

Python Arithmetic operators

//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 = 4 and 9.0//2.0 = 4.0



PRIME INTUIT
Finishing School

Python Arithmetic operators

```
a, b, c = 21,10,0
c = a + b
print ("Line 1 - Value of c is ", c)
c = a - b
print ("Line 2 - Value of c is ", c )
c = a * b
print ("Line 3 - Value of c is ", c)
c = a / b
print ("Line 4 - Value of c is ", c )
c = a % b
print ("Line 5 - Value of c is ", c)
a = 2
b = 3
c = a**b
print ("Line 6 - Value of c is ", c)
a = 10
b = 5
c = a//b
print ("Line 7 - Value of c is ", c)
```

PRIME INTUIT
Finishing School

Python Comparison Operators

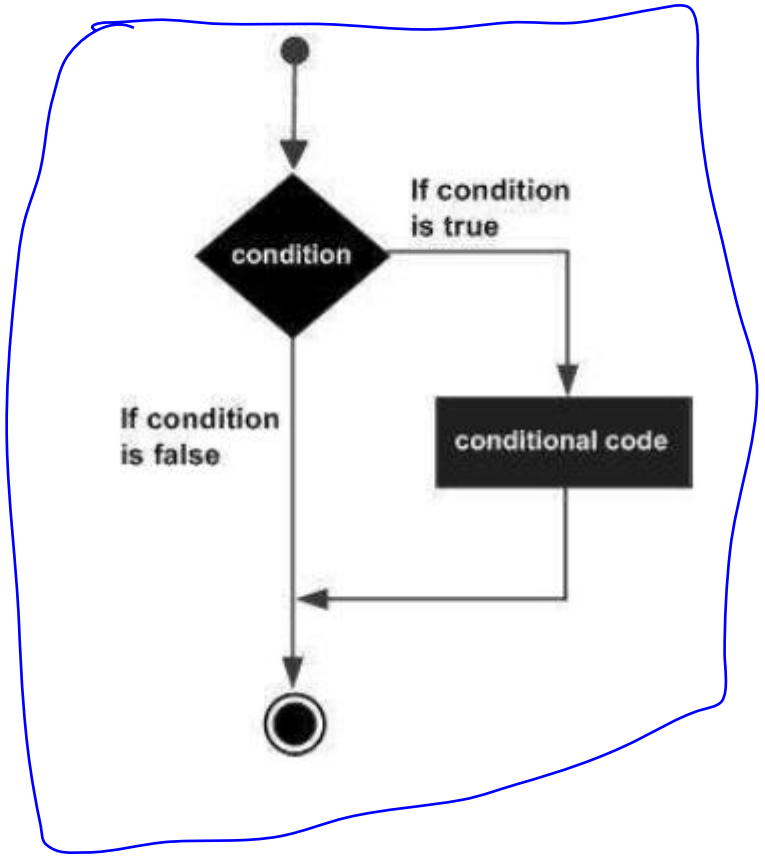
Assuming a = 10 and b = 21

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a!= b) is true.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

PRIME INTUIT
Finishing School

Python If Statement

The IF statement is similar to that of other languages. The **if** statement contains a logical expression using which the data is compared and a decision is made based on the result of the comparison.



Syntax:

```
if Condition:
        Condition code
```

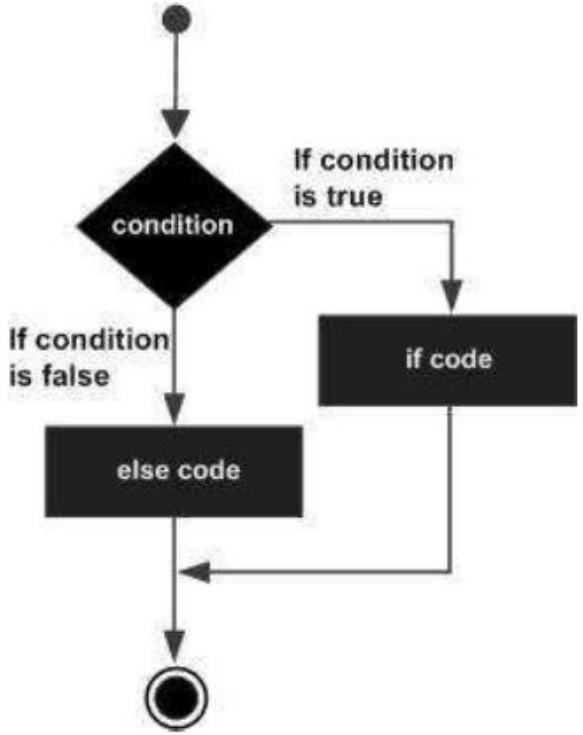
example:

```
if a > b:
        print(a)
print(b)
```

PRIME INTUIT
Finishing School

Python If ……Else Statement

An **else** statement can be combined with an **if** statement. An **else** statement contains a block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.
The else statement is an optional statement and there could be at the most only one **else** statement following **if**.



Syntax:

```
if Condition:
        Condition code
else:
        Condition code
```

PRIME INTUIT
Finishing School

Python If ……Else Statement Example

Your working in an it department of a mega store, the store wants to run a discount of 5% if the billing amount is less then Rs 1000 and 10% discount if the billing value is greater then Rs 1000, you can except the billing amount as an input to begin with.

```
amount=int(input("Enter amount: "))
if amount<1000:
discount=amount*0.05
print ("Discount",discount)
else:
discount=amount*0.10
print ("Discount",discount)
print ("Net payable:",amount-discount)
```

PRIME INTUIT
Finishing School

Python If ……Else Statement Example

Write a program to calculate the electricity bill (accept number of unit from user) according to the following criteria:

Unit	Price no
First 100 units	No Charge
Next 100 units	Rs 5 per unit
After 200 units	Rs 10 per unit

(For example, if input unit is 350 than total bill amount is Rs2000)

Hint : if else statements

```
Units=int(input("Enter number of Units: "))
```

Python Comparison Operators Examples

```
a = 21, b = 10
if ( a == b ):
print ("Line 1 - a is equal to b")
else:
print ("Line 1 - a is not equal to b")
if ( a != b ):
print ("Line 2 - a is not equal to b")
else:
print ("Line 2 - a is equal to b")
if ( a < b ):
print ("Line 3 - a is less than b" )
else:
print ("Line 3 - a is not less than b")
```

PRIME INTUIT
Finishing School

Python Comparison Operators

```
if ( a > b ):
print ("Line 4 - a is greater than b")
else:
print ("Line 4 - a is not greater than b")
a,b=b,a #values of a and b swapped. a becomes 10, b becomes 21
if ( a <= b ):
print ("Line 5 - a is either less than or equal to b")
else:
print ("Line 5 - a is neither less than nor equal to b")
if ( b >= a ):
print ("Line 6 - b is either greater than or equal to b")
else:
print ("Line 6 - b is neither greater than nor equal to b")
```

PRIME INTUIT
Finishing School

add sir's updated slides

Python operators Precedency

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Ccomplement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators

PRIME INTUIT
Finishing School

Thank You !