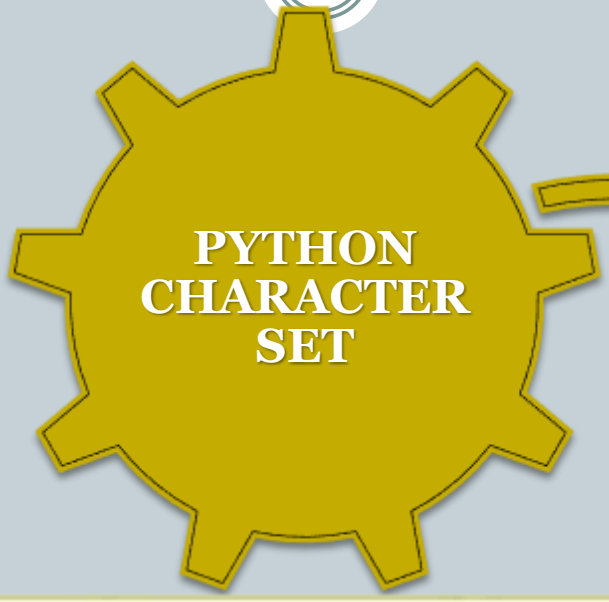
PYTHON FUNDAMENTALS

PYTHON CHARACTERSET



- **Letters:- A-Z, a-z**
- **Digits:- 0 to 9**
- **Special Symbols:- space + - / ( ) [ ] = ! = < > , ‘ “ $ # ; : ? &**
- **White Spaces:- Blank Space , Horizontal Tab, Vertical tab, Carriage Return.**
- **Other Characters:- Python can process all 256 ASCII and Unicode Characters.**

1. Keyword/Reserved Word

What is Keyword?

Keywords are also called as reserved words these are having special meaning in python language. The words are defined in the python interpreter hence these cant be used as programming identifiers.

fppt.com

Some Keywords of Python Language

and	assert
break	class
continue	def
del	elif
else	except
exec	finally
for	from

fppt.com

PYTHON NAMING CONVENTIONS

SOME VALID IDENTIFIERS:

Myfile1 DATE9_7_8

y3m9d3 _xs

MYFILE _FXd

SOME INVALID IDENTIFIERS:

MY-REC 28dre break

elif false del

fppt.com

Variables in Python

- A variable has
  - A name – identifier
  - A data type - int, float, str, etc.
  - Storage space sufficient for the type.

LITERALS / CONSTANT VALUES

What is literals?

Literals are also called as constants or constant values these are the values which never change during the execution of program.

Numeric Data Types

- int

  This type is for whole numbers, positive or negative. Examples: 23, -1756

- float

  This type is for numbers with possible fraction parts. Examples: 23.0, -14.561

Integer operators

The operations for integers are:

+	for addition
-	for subtraction
*	for multiplication
/	for integer division: The result of 14/5 is 2
%	for remainder: The result of 14 % 5 is 4

- *, /, % take precedence over +, -
  x + y * z will do y*z first

- Use parentheses to dictate order you want.
  (x+y) * z will do x+y first.

Integer Expressions

- Integer expressions are formed using
  - Integer Constants
  - Integer Variables
  - Integer Operators
  - Parentheses

Python Assignment Statements

- In Python, = is called the *assignment operator* and an *assignment statement* has the form

  <variable> = <expression>

- Here
  - <variable> would be replaced by an actual variable
  - <expression> would be replaced by an expression

- Python: age = 19

Python Assignment Statement

- Syntax: <variable> = <expression>
  - Note that variable is on left

- Semantics:

  Compute value of expression

  Store this as new value of the variable

- Example: Pay = PayRate * Hours

10	40	400
Payrate	Hours	Pay

What about floats?

- When computing with floats, / will indicate regular division with fractional results.
- Constants will have a decimal point.
- 14.0/5.0 will give 2.8 while 14/5 gives 2.

Comments

- Often we want to put some documentation in our program. These are comments for explanation, but not executed by the computer.

- If we have # anywhere on a line, everything following this on the line is a comment – ignored

Numerical Input

- To get numerical input from the user, we use an assignment statement of the form

  <variable> = input(<prompt>)

- Here
  - <prompt> would be replaced by a prompt for the user inside quotation marks
  - If there is no prompt, the parentheses are still needed
- Semantics
  - The prompt will be displayed
  - User enters number
  - Value entered is stored as the value of the variable

INPUT ( ) FUNCTION

int ( ) and float ( ) Functions:

Python offers two functions to be used with input( ) to convert the received values:

Example 1: >>age = int(input("Enter age"))

Example 2: >>sal=float(input("Enter salary))

fppt.com

Print Statement

- For output we use statements of the form

  print <expression>

- Semantics
  - Value of expression is computed
  - This value is displayed

- Several expressions can be printed – separate them by commas

Example - Fahrenheit to Centigrade

- We want to convert a Fahrenheit temperature to Centigrade.
- The formula is $C = (F - 32) \times 5/9$
- We use type float for the temperatures.

String Concatenation

- To concatenate, or combine, two strings you can use the + operator.
- Example
- Merge variable a with variable b into variable c:
- a = "Hello"
  b = "World"
  c = a + b
  print(c)

Python Operators

- Python divides the operators in the following groups:
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators
- https://www.w3schools.com/python/python_operators.asp

Relational Operators

- Many logical expressions use *relational operators*:

Operator	Meaning	Example	Result
==	equals	1 + 1 == 2	True
!=	does not equal	3.2 != 2.5	True
<	less than	10 < 5	False
>	greater than	10 > 5	True
<=	less than or equal to	126 <= 100	False
>=	greater than or equal to	5.0 >= 5.0	True

Logical Operators

- These operators return true or false

Operator	Example	Result
and	9 != 6 and 2 < 3	True
or	2 == 3 or -1 < 5	True
not	not 7 > 0	False

DANGER! Operator Overloading!

- NOTE! Some operators will work in a different way depending upon what their operands are. For example, when you add two numbers you get the expected result: `3 + 3` produces 6.
- But if you “add” two or more strings, the + operator produces a concatenated version of the strings: `“Hi” + “Jay”` produces “HiJay”
- Multiplying strings by a number repeats the string!

`“Hi Jay” * 3` produces “Hi JayHi JayHiJay”

- The % sign also works differently with strings:

`“test %f” % 34` produces “test 34”

Calculate Kinetic Energy

```
print("This program calculates the kinetic energy of a moving object.")
m =float(input("Enter the object's mass in kilograms: "))
v = float(input("Enter the object's speed in meters per second: "))
e = 0.5 * m * v * v
print("The object has " + str(e) + " joules of energy.")
```