Problem Solving and Programming

Prof. Tulasi Prasad

August 2, 2019

- Operators and Expressions in Python
  - **Basic Arithmetic Operations in Python**

command	Name	Example	Output
+	Addition	4 + 5	9
-	Subtraction	8 - 5	3
*	Multiplication	4 * 5	20
/	True Division	19 / 3	6.333
//	Integer Division	19 // 3	6
%	Remainder	19 % 3	1
**	Exponent	2 ** 4	16

- Operators and Expression in Python
  - **Order of Operations:**
    - 1. Parentheses ()
    - 2. Exponents **
    - 3. Multiplication *, Division / and remainder %
    - 4. Addition + and subtraction -

Operator	**Operation**	**Precedence**	**Associativity**
()	Parentheses	0	Left to Right
**	Exponentiation	1	Right to Left
*	Multiplication	2	Left to Right
/	Division	2	Left to Right
//	Integer Division	2	Left to Right
%	Remainder	2	Left to Right
+	Addition	3	Left to Right
-	Subtraction	3	Left to Right

- **Operators and Expressions in Python**
  - **Examples:**
    - >>> 1 + 2 * 3 → check output → Priority
    - >>> (1 + 2) * 3 → check output → Priority
    - >>> 4 - 40 - 3 → check output → Associativity
    - >>> 4 -(40 - 3) → check output → priority
    - Check the following outputs in interactive mode of python
    - word = 'word' → check output
    - sentence = "This is Sentence" → check output
    - paragraph = """ This is a paragraph. it is made of multiple line""" → check output → legal
    - name = int(input('Enter name')) → given name as vit → check output → reason

- **Built - in Format Function:**
  - Built-in Function is used to produce a numeric string version of a value containing a specific number.
  - >>> 12/5 → check output → 2.4
  - >>> format(12/5, '.2f') → check output
  - >>> 5/7 → check output
  - >>> format(5/7, '.2f') → check output → '.2f' rounds the result to two decimal places of accuracy in the string produced.
  - For very large and very small values 'e' can be used as a format specifier.
  - >>> format(2 ** 100, '.6e') → check output
  - >>> format(11/12, '.2f') → check output
  - >>> format(11/12, '.3f') → check output
  - >>> format(11/12, '.2e') → check output
  - >>> format(11/12, '.3E') → check output

- **Python: Dynamic Type Language**
  - Same variable can be associated with values of different type during program execution.
  - Example
    - >>> var10 = 10
    - >>> var10 = 10.24
    - >>> var10 = ' VIT '
  - It is also very dynamic as it rarely uses what it knows to limit variable usage

- **Examples**

LET'S TRY IT

From the Python Shell, enter the following and observe the results.

```
>>> num = 10
>>> num
???
>>> id(num)
???

>>> num = 20
>>> num
???
>>> id(num)
???

>>> k = num
>>> k
???
>>> id(k)
???
>>> id(num)
???
```

```
>>> k = 30
>>> k
???
>>> num
???
>>> id(k)
???
>>> id(num)
???

>>> k = k + 1
>>> k
???
>>> id(num)
???
>>> id(k)
???
```

- **Bitwise Operators:**
  - Manipulation can be done at bit level.
  - It treats the integers as strings of binary bits.
  - These operators are useful in network packets, serial port and binary packet data.
  - >>> x = 1 → decimal 1 is 0001 in bits
  - >>> x << 2 →check output → Shift left 2 bits → 0100
  - Three different bitwise operators:
    - 1. Bitwise AND Operator
    - 2. Bitwise OR Operator
    - 3. Bitwise NOT Operator
  - >>> x | 2 → check output → Bitwise OR Operator
  - >>> x & 2 → check output → Bitwise AND Operator
  - >>> x = 0b0001 → >>> bin(x << 2) → check output
  - >>> bin(x | 0b010) → check output
  - >>> bin(x & 0b1) → check output

- **Logical Operators:**
  - Assume a = 10 and b = 20

Logical Operator	**Description**	**Example**
and	If both the operands are true then condition becomes true	(a and b) is true
or	If any of the two operands are non-zero then condition becomes true	(a or b) is true
not	Used to reverse the logical state of its operand	Not(a and b) is false

- **Python is strongly typed language:**
  - interpreter keeps track of all variables types
  - Check type compatibility while expressions are evaluated
  - >>> 2 + 3 → check output
  - >>> 'two' + 1 → check output
- **Shorthand Assignment operators**

Table 11-2. Augmented assignment statements

X += Y	X &= Y	X -= Y	X |= Y
X *= Y	X ^= Y	X /= Y	X >>= Y
X %= Y	X <<= Y	X **= Y	X //= Y

- Python program for Bob Problem

```
n = float(input("Enter amount in hand"))
c = float(input("Enter price of one chocolate"))
m = int(input("Enter num of wrapper for free chocolate"))
#compute number of chocolates bought
p = n//c
#compute number of free chocolates
f = p//m
print("Number of chocolates",int(p+f))
```

Problem - 1

ABC company Ltd. is interested to computerize the pay calculation of their employee in the form of Basic Pay, Dearness Allowance (DA) and House Rent Allowance (HRA). DA and HRA are calculated as certain % of Basic pay(For example, DA is 80% of Basic Pay, and HRA is 30% of Basic pay). They have the deduction in the salary as PF which is 12% of Basic pay. Propose a computerized solution for the above said problem.

- We know the **PAC chart** and **Flowchat**

CODE

```
#Enter the basic pay
bp=float (input('Enter the basic pay:'))
# net pay calucluation
netpay =bp + (bp*0.8) + (bp*0.3) - (bp*0.12)
# display net salary
print ('Net pay:', netpay)
```

- **lambda Operator:**
  - The lambda operator or lambda function is a way to create small anonymous functions → i.e. functions without a name.
  - Example
  - >>> ftoc = lambda f:(f-32)*5.0/9
  - >>> print(ftoc(104)) → check output