Introduction to Programming

Department of Computer Science and Information Systems

Lecturer: Steve Maybank

sjmaybank@dcs.bbk.ac.uk

Autumn 2019 and Spring 2020

Week 3: Arithmetic and Built in Functions

Overview

- Review of week 2:
  - number types
  - creating variables
- Arithmetical problems
- Arithmetic
- Built in functions
- See Python for Everyone, Ch. 2.2

Recall Number Types

- Type int: whole numbers, positive, zero and negative, e.g. 1, 0, -1.
- Type float: decimal fractions, e.g. 1.28, 3.0, -2.5, 0.0.
- Warning: a number literal such as 1.3 cannot be represented exactly in a running program. Instead, the program uses a number of type float very near to 1.3.

Recall Creation and Initialisation of Variables

```
cansPerPack = 6
# create the variable cansPerPack and
# initialise it with the value 6 of type int

cansPerPack = 8
# overwrite the previous value 6 with the new value 8

cansPerPack = 8.5
# The values of cansPerPack can be switched from type
# int to type float and conversely
```

Square Root of 2

- Let x be a number such that x^2 is near to 2

- Let δ be a small number such that $(x + \delta)^2 = 2$. Then

$$2 = (x + \delta)^2 = x^2 + 2x\delta + \delta^2 \approx x^2 + 2x\delta$$

- Solve (approximately) for δ to obtain

$$\delta \approx \frac{1}{x} - \frac{x}{2} \qquad \text{and} \qquad x + \delta \approx \frac{x}{2} + \frac{1}{x}$$

Square Root of 2

```
>>> x=1.4
>>> x**2
1.9599999999999997
>>> x=x/2+1/x
>>> x
1.4142857142857141
>>> x**2
2.0002040816326527
>>> x=x/2+1/x
>>> x
1.4142135642135643
>>> x**2
2.000000005205633
>>> x=x/2+1/x
>>> x
1.414213562373095
>>> x**2
1.9999999999999996
```

- Let x be a number such that x^2 is near to 2
- Let δ be a small number such that $(x+\delta)^2 = 2$. Then

$$2 = (x+\delta)^2 = x^2 + 2x\delta + \delta^2 \approx x^2 + 2x\delta$$

- Solve (approximately) for δ to obtain

$$\delta \approx \frac{1}{x} - \frac{x}{2} \qquad \text{and} \qquad x + \delta \approx \frac{x}{2} + \frac{1}{x}$$

Find square root of 2:

Begin with an approximation x to the square root of 2
Find a better approximation x + δ
The process iterates, with x + δ in place of x

Problem Statement

- You have the choice of buying two cars.
- One is more fuel efficient than the other, but also more expensive.
- You know the price and fuel efficiency (in miles per gallon, mpg) of both cars.
- Assume a price of £4 per gallon of petrol and usage of 15,000 miles per year.
- You plan to keep the car for 10 years.
- You will pay cash for the car and not worry about financing costs.
- Which car is the better deal?

Problem Break Down

- 1st stage

  Find the total cost for each car

  Choose the car that has the lowest total cost

- 2nd stage: total cost of a car (£) = purchase price (£) + operating cost (£)

- 3rd stage: operating cost (£)

  = number of years to run * annual fuel cost (£)

- 4th stage: annual fuel cost (£)

  = price per gallon (£/gal) * annual fuel consumed (gal)

- 5th stage: annual fuel consumed (gal)

  = annual miles driven (miles) / fuel efficiency (miles/gal)

Description of Each Step

- The descriptions are in pseudocode. The steps are ordered such that each step can be carried out using the results of previous steps

- for each car, compute the total cost as follows

  annual fuel consumed = annual miles driven/fuel efficiency

  annual fuel cost = price per gallon x annual fuel consumed

  operating cost = number of years to run x annual fuel cost

  total cost = purchase price + operating cost

  *All the values of the cyan coloured variables are known/given.

- if total cost 1 $<$ total cost 2 choose car 1 else choose car 2

Example

- R1.15. You want to decide whether you should drive your car to work or take the train.
- You know the distance from your home to your place of work, and the fuel efficiency of your car (in miles per gallon). The cost of petrol is £4 per gallon and car maintenance is 20p per mile.
- You also know the price of a return train ticket.
- Write an algorithm to decide which commute is cheaper.

Example

- Compare drive to work and take the train

- Drive to work

driveToWork = (carMaintenanceCostPerMile +fuelCostPerMile) * distanceHomeWork * 2

fuelCostPerMile = fuelCostPerGallon / fuelEfficiency(miles per gal)

- Take the train

priceReturnTicket

Operators, Variables and Literals

- Operators act on one or more numbers to produce a new number, e.g.

  Addition +

  Subtraction –

  Multiplication *

  Division /

- Variables

  p, q, cansPerPack, …

- Number literals

  4, 5, -64.8, 27.305, …

Expressions

- An expression is a combination of operators, variables, literals and parentheses

- Examples

  (3+4)/2

  p+p*p

  (p+p)*p

  3+4+5

  3-4-5

Arithmetic Operators in Python

- Addition:

  ```
  p = 3+4      # assign value of type int
  q = 3.1+7  # assign value of type float
  ```

- Subtraction:

  ```
  p = 3-4
  q = 4.89-1.7
  ```

- Multiplication

  ```
  p = 4*5      # other versions , e.g. 4x5, 4•5 not permitted
  q = 4.0*5  # assign value of type float
  ```

Power Operation

- Power operation:

```
p = 5**2          # assign 25 = 5*5
q = 10**2**3      # assign 10**(2**3) = 10^8
```

$$q = 10^{8}$$

** is evaluated from right to left

** has a higher precedence over other operators

Write results for 10*2**3, 10**2**3**2, 10**2**3*4 and 10**4*3**2

- Exercise: write out the Python code for

$$b \times \left(1 + \frac{r}{100}\right)^{n}$$

Division

16 ÷ 3 = 5.3333333333333333

dividend divisor decimal quotient

---

16 ÷ 3 = 5 R 1

dividend divisor (int) quotient remainder

---

dividend = quotient x divisor + remainder

16 = 5 x 3 + 1

Division

16 ÷ 3 = 5.33333333333333333

dividend divisor decimal quotient

---

16 ÷ 3 = 5 R 1

dividend divisor (int) quotient remainder

-16 ÷ 3 = ? R ?

Division

16 ÷ 3 = 5.3333333333333333

dividend divisor decimal quotient

---

16 ÷ 3 = 5 R 1

dividend divisor (int) quotient remainder

-16 ÷ 3 = -6 R 2

always of the same sign

dividend divisor quotient remainder

Division and Remainder Operators

- Division (/): the result is the decimal quotient

  p = 6/4   # assign value 1.5 of type float

  q = 2.1/-7 # assign value -0.3 of type float

- Floor division (//): the result is the quotient

  # always round down the decimal quotient

  p = 6//4   # round down to 1

  q = (-6)//4 # round down to -2

  # avoid using // with arguments of type float

6//-4=?	-1.5 round down to -2
(-6)//(-4)=?	1.5 round down to 1

- Remainder (%) : the result is the remainder

  # always has the same sign as divisor

  p = 5%4   # remainder 1 on dividing 5 by 4

  q = (-5)%4 # see next slide

Remainder Operator

- Remainder (%) : the result is the remainder (always the same sign as divisor)

q = (-5)//4 #What is the value of q?

What is the remainder of (-5) ÷ 4?

(-5) // 4 = -2 (-1.25 round down to -2) the quotient is -2

dividend = divisor x quotient + remainder

- 5 = 4 x (-2) + remainder, so remainder = 3

(-5) ÷ 4 = (-2) R 3, so (-5)%4 = 3

17%5=?
(-17)%5=?
17%(-5)=?
(-17)%(-5)=?

Remainder Operator

- Remainder (%) : the result is the remainder (always the same sign as divisor)

  q = (-5)//4 #What is the value of q?

  # What is the remainder of (-5) ÷ 4?

  # (-5) // 4 = -2 (-1.25 round down to -2) the quotient is -2

  # dividend = divisor x quotient + remainder

  # - 5 = 4 x (-2) + remainder, so remainder = 3

  # (-5) ÷ 4 = (-2) R 3, so (-5)%4 = 3

17%5=3R2
(-17)%5=(-4)R3
17%(-5)=(-4)R(-3)
(-17)%(-5)=3R(-2)

Table for Floor Division and Remainder

For n = 1729

Check the web for the significance of 1729

Expression	Value	Comments
n%10	9	For any positive integer n, n%10 is the last digit of n
n//10	172	This is n without the last digit
n%100	29	The last two digits of n
n%2	1	n%2 is zero if n is even and 1 if n is odd
-n//10	-173	-173 is the largest integer <= -172.9

Decimal Digits

- The operators // and % can be used to extract the digits of a decimal integer
- Examples

  385%10 = 5

  385//10 = 38

  (385//10)%10 = 8
- To extract from an integer n the ith digit from the right, use

  (n//(10**(i-1)))%10

  987654321 How to extract 6?

  735502646188 How to extract the two 6?

More About Expressions

- Literals and names of variables are expressions
- If e, f are expressions then
  (e)+(f), (e)-(f), (e)*(f), (e)/(f), (e)//(f), (e)%(f), (e)**(f) are expressions
- Examples: 4, 5, p are three expressions,
  - therefore (p)+(4) is an expression,
  - therefore (5)*((p)+(4)) is an expression, and so on

Precedence

- The number of brackets needed in an expression is reduced by specifying a precedence for the operators

- Exponentiation ** takes precedence over
  multiplication *, real division /, remainder %,
floor division //,
  which in turn take precedence over
  addition + and subtraction -

Examples of Precedence

**	>	*, /, %, //	>	+, -

```
p = 4*3+1          # value ?
p = 3*2**3         # value ?
p = 3.0*5.5/6.0    # value ?
p = 1+3*2          # value ?
```

See PFE Appendix B. If in any doubt use brackets

Built in Function abs

- The function abs takes a number as an argument and returns the absolute value of that number, e.g.

distance1 = abs(-5)

the argument of abs is -5, the value 5 is returned

distance2 = abs(5)

the argument of abs is 5, the value 5 is returned

Additional Built in Functions

- round(x): return the value of x rounded to a whole number

  p = round(1.6)   # assign the value 2

  q = round(n+0.5)

  # n is an even number, round(n+0.5)=n
  # n is an odd number, round(n+0.5)=n+1

  #round(3.5)=4, round(2.5)=2

  Always end up with an even number!

- round(x, n): return the value of x rounded to n decimal places

  p = round(1.572, 2) # assign the value 1.57

- max(x, y, z, …): return the largest value of the arguments
- min(x, y, z, …): return the smallest value of the arguments

Associativity

- All operators have left to right associativity except exponentiation which has right to left associativity

```
p=3-4-7     # value
p=1/2/4     # value
p=2**2**3 # value
p=4+5+6   # rule for associativity not required
```

See PFE Appendix B. If in any doubt use brackets. Never write anything as horrible as 1/2/4

Question R2.5

- What are the values of the following expressions? In each line assume

  x = 2.5   y = -1.5   m = 18   n = 4

  1. x*y-(x+n)*y
  2. m//n+m%n
  3. 5*x-n/5
  4. 1-(1-(1-(1-(1-n))))
  5. sqrt(sqrt(n))   # sqrt is the square root function

Question R2.6

- What are the values of the following expressions, assuming that n is 17 and m is 18?

1. n//10+n%10
2. n%2+m%2
3. (m+n)//2
4. (m+n)/2.0