#1.py

# Somebody tells you today's temperature in degrees Celsius (a scale).
# You want to convert it to degrees Fahrenheit (a different scale),
# conversion formula: \( f = 32 + \frac{9}{5}c \)

# Algorithm (high-level but precise solution in steps). Algorithm describes
# the procedure you must follow.

# Obtain \( c \), the degrees Celsius
# Do the conversion \( f = \left(\frac{9}{5}\right)c + 32 \)       [Note * means multiplication]
# Output the result \( f \)

# Pseudo-code is an informal language that helps you develop algorithms
# and programs. Once you learn a programming language, your pseudo-code
# may look a lot like that language, except it will be informal, since the
# interpreter or compiler never gets to see it directly (and complain) :)

# Now the actual Python code [only 4 lines really, the rest are comments] :)

# Program to convert degrees Celsius to degrees Fahrenheit
# author: Harry Hopeitrunz

def main():
    c = eval(input("Please tell me the temperature in degrees Celsius: "))

    # If you don't use eval, a character string representing the temperature
    # will be input in c. Because we want to do an arithmetic conversion, we need
    # c to be understood as a number by Python
    f = \left(\frac{9}{5}\right)c + 32

    # c, f are called variable names (or just variables, or identifiers)
# remember that capital C is different from lower-case c

```python
print("The temperature is ", f," degrees Fahrenheit")
```

# Now run the program and test it!

#Every time you write a program you must TEST IT with as much data as you can to exercise different parts of it (because it may be large/complex)

#If it does not run, you have a syntax error (language problem) or a semantic error (logic problem). Now you must DEBUG your program.

#Important TIP: Always plan before writing code. Be careful enough that your program will run without errors, or at most small errors you catch immediately. Otherwise, when you start to DEBUG you have begun to solve a new problem (what went wrong?) and not the orginal programming problem!

# To test the results you can use your calculator or the following link

# https://www.google.com/?gws_rd=ssl#q=convert+celsius+to+fahrenheit

# but remember it may sometimes happen that the program you are checking against is also a program that gives the wrong result!

# So you must be sure of the correct solution.

#2.py

#variables need to be assigned values, or else Python sees them as undefined

```python
def main():
    x = 99

    print(x)  # x is defined, so no problem here

    print(y)  # y is undefined, and Python will complain. Check this out!

    print(Harry)  # what is Harry? It's a variable. But what is it's value?

    #The Python interpreter will complain about y being undefined and stop running the function at that point, because the program is faulty

    #If you were using a compiler instead, it would process the whole program
    #and tell you all the syntax errors
```

#3.py
# You can work with all kinds of arithmetic expressions, keeping in mind
# that the usual rules of precedence hold.

# Precedence Rules (top to bottom)
#  Highest precedence
#  ( )      (anything in brackets is done first)
#  **     (exponentiation)
#  -x, +x
#  *, /, %, //
#  +, -
#  relational operators: <, >, <=, >=, !=, ==
#  logical not
#  logical and
#  logical or
#  Lowest precedence

def main():
    x = 11/2 * 5 + 3  # 11/2 is 5.5, times 5 gives 27.5, adding 3 gives 30.5
    print(x)

    # all spaces are ignored in arithmetic expressions
    y = 3 + 11/2 * 5  # result is the same, 30.5
    print(y)  # "/" and "*" have higher precedence than "+", so done first

#Remember: if you want to be clear, use parentheses in expressions
#Otherwise, Python will use the precedence rules

#  % is the modulus operation; it means divide and give the remainder
    z1 = 11%2  # answer is 1, since remainder is 1
    print(z1)

    print( 27 % 5 * 10/2 + 3 )  # answer is (2 * 5) + 3 = 13
    # notice how we did not even use a variable

    #Now observe how the parenthesis has highest precedence
    z2 = (11 + 12) - 3 * 3  # 11 + 12 is done first, answer is 14
    print(z2)

    #exponentiation in an expression
    z3 = 2 * 3**3  # 3**3 is 3 raised to power of 3, so 27; ans is 54
print(z3)

# variables and assignments

def main():
    # You can change the value of any variable anytime
    Jane = "Larry"  # Larry is a character string
    print(Jane)
    Jane = "Tarzan"  # Jane's value is now Tarzan, and not Larry
    print(Jane)
    Jane = 29
    print(Jane)  # Jane's value is now 29

    # You can think of Jane as pointing to a new memory location every time
    # you assign a new value. It loses the association with the old location
    # to which it was previously pointing

    # Suppose x = 5, y = 6 and you want to swap their values; in a language
    # like C or Java you would do it in this simple way which also works here

    x = 5
    y = 6
    print("Before swap: x = ", x, " y = ", y)

    # now swap values
    temp = x  # keep x's value somewhere so it does not get clobbered
    x = y  # when you put y's value into it
    y = temp  # then move the value from that somewhere into y
             # otherwise both x and y will end up with value 6

    print(" After swap: x = ", x, " y = ", y)

    # But Python gives you a shortcut too!
```python
x = 5
y = 6
print("Before swap: x = ", x, ", y = ", y)

x, y = y, x  # x gets value y, and y gets value x, simultaneously
print(" After swap: x = ", x, ", y = ", y)

# So in a single assignment statement you can do many things
x = 2
y = 3

add, mul = x+y, x*y

print( " x = ", x," y = ", y," sum = ", add," product = ", mul)

#Loops are used to execute segments of code repeatedly

# Let's use a loop to find the sum of the first 100 integers
# We already know the answer. It's n*(n+1)/2, so we use it to check.

def main():
    s = 0
    # i starts at 0 and stops at 100 in the for-loop
    for i in range(101):
        s = s + i
        print (s)

# Now suppose you have $100 in your savings account at PEFCU
# and just suppose that PEFCU gives you an interest of 7.2% per year on
# your savings account balance

# at the end of the first year you will have 100*(1 + 0.072) = $107.20
# at the end of the second year you will have 107.20*(1 + 0.072) = ?
# Let's find out how much you have after n years

s = 100     # your initial account balance
```
```python
n = eval(input("How many years will you leave this money untouched? "))

for i in range(n):
    s = s*(1 + 0.072)
    print("At the end of year ",i+1," account balance = ",s)

# If you started with $1000 at the same rate of interest, how long would it take
# you to become a millionaire?

#--------------------------------------------------

From:
http://courses.cs.purdue.edu/- Computer Science Courses

Permanent link:
http://courses.cs.purdue.edu/cs17700:fall:week1_examples2

Last update: 2015/08/26 21:20