Part I: Types, variables and comments

Python

```python
# commentaar
''' multi-line commentaar '''

x = 5                    # type is dynamic


# strings



bedrijf = "Python inc." # string
bedrijf = 'Python inc.' # alternatieve syntax




# floats en wiskundige expressies
farenheit = 50.1         # float


y = 7.5
x = x + 1                # x is 6



x = 0.1 * x              # x is 0.5 (en float)



import math              # import module <math>
from math import pi,cos # cos  == math.cos
x = 2**3                 # x^3 is 8

x = 2//3                 # x is 0 (gehele deling)


x = 5 % 3                # x is 2, de rest bij
                         # deling (of modulo)
x = 2.0/3.0              # x is 0.6666666667

x = 0.5 * (x + y)        # x is 6.25

x = 2 * math.pi          # x is +- 6.28
x = math.cos(x)          # x is 1.0

x = cos(2 * pi)          # x is 1.0

celcius = 5.0/9.0 * (farenheit - 32)

# printing



s = "Hallo"
print(s)

# "%s" substitutie "Python Inc."
print("Welkom bij deze %s programmeertaal" %
bedrijf)
```

C++

```cpp
//commentaar
/* multi-line commentaar */

x = 5;                   //fout: geen declaratie type
int x = 5;               //correct: int, type is static

// strings
#include <string>       //import module <string>
using namespace std;     //string == std::string

string bedrijf = "C++ inc.";
          // ' mag enkel gebruikt voor characters

// alternatief is C string:
// e.g. zelfde als ['C','+','+',' ', 'i','n','c','.',\0]
char bedrijfCString[] = "C++ inc.";

# floats en wiskundige expressies
double farenheit = 50.1;      //typisch 64 bits
float farenheit_flt = 50.1;   //typisch 32 bits

double y = 7.5;
x = x + 1;               // x is 6
x++;                     // alternatieve syntax
x+=1;                    // alternatieve syntax

x = 0.1 * x;             // x is 0, want int dus
                         //decimale gedeelte gaat weg
double xFloat = 0.1 * x; // xFloat is 0.5

#include <cmath >        //bevat pow, sqrt, cos etc.
                         //uit C library
xFloat = pow(2.0,3.0);   // x^3 is 8

x = 2/3;                 // x is 0, omdat 2 en 3
                         // integers zijn en omdat x
                         // een integer is
x = 5 % 3;               // % x is 2, de rest bij
                         // deling (of modulo)
xFloat = 2.0/3.0;        // xFloat is 0.6666666667
                         // want lhs en/of rhs double
xFloat = 0.5 * (x + y);  // xFloat is 6.25

xFloat = 2 * M_PI;       // xFloat is +- 6.28
xFloat = cos(x);         // xFloat is 1.0

xFloat = cos(2 * pi);    // x is 1.0

double celcius = 5.0/9.0 * (farenheit - 32);

# printing
#include <iostream>   //bevat cout, cin
using namespace std;  // (std::)cout, (std::)string

string s = "Hallo";
cout << s << endl;

// substitutie "C++ Inc."
cout << "Welkom bij deze " << bedrijf
     << " programmeertaal" << endl;
```

Part 2: List/Vector

Python

```
# lijst van ints:
mijn_lijst = [1,3,7,99,34,20]
# lengte is 6
lengte = len(mijn_lijst)



print("Lijst is %d groot" % lengte)









# lijst begint op index 0, en toegang met []
eerste_waarde = mijn_lijst[0]
print("Eerste is %d" % eerste_waarde)

# lijst eindigt op lengte-1
laatste_waarde = mijn_lijst[ len(mijn_lijst)-1]
print("Laatste is %d" % laatste_waarde)

# wijzig lijst
mijn_lijst[1] = 13      # 3 wordt 13
mijn_lijst[10]          # out-of-range fout




# voeg element toe aan lijst
mijn_lijst.append(10)   # [1,13,7,99,34,20,10]
mijn_lijst.append(5)    # [1,13,7,99,34,20,10,5]
mijn_lijst.append(23)   # [1,13,7,99,34,20,10,5,23]

# verwijder op index
del mijn_lijst[0]       # [13,7,99,34,20,10,23]
del mijn_lijst[6]       # [13,7,99,34,20,10]
```

C++

```
//C array met vaste grootte:
int mijn_lijst_arr[] = {1,3,7,99,34,20};
//array heeft geen functie om zijn eigen vaste lengte
//te bepalen, dus moet je dit zelf bijhouden en
//lengte als afzonderlijk argument doorgeven aan functie
int lengte = 6;


cout << "Array is " << lengte << " groot" << endl;

//alternatief (en beter) werk je met vector, hetgeen net
//zoals een list een dynamische capaciteit heeft en
//een eenvoudige gebruik
#include <vector>
using namespace std;
vector<int> mijn_lijst = {1,3,7,99,34,20};
cout << "Lijst is " << mijn_lijst.size() << " groot"
     << endl ;

// vector begint op index 0 en toegang met []
int eerste_waarde = mijn_lijst[0];
cout << "Eerste is " << eerste_waarde << endl;

// vector eindigt op size -1
int laatste_waarde = mijn_lijst[mijn_lijst.size()-1];
cout << "Laatste is " << laatste_waarde << endl;

// wijzig vector
mijn_lijst[1]= 13;   //3 wordt 13
mijn_lijst[10];  // "soms" fout
                 // door [] te gebruiken veronderstelt
                 // C++ dat je direct in het geheugen
                 // zonder check, een waarde haalt
mijn_lijst.at(10);   //out-of-range-fout

//voeg element toe aan vector
mijn_lijst.push_back(10);  // [1,13,7,99,34,20,10]
mijn_lijst.push_back(5);   // [1,13,7,99,34,20,10,5]
mijn_lijst.push_back(23);  // [1,13,7,99,34,20,10,5,23]

//verwijder op index
mijn_lijst.erase(mijn_lijst.begin());
mijn_lijst.erase(mijn_lijst.begin()+6);
// begin() is een iterator, die verwijst naar
// het eerste element
```

Part 3: Dictionary/Map

Python

```
# Voorbeeld met dictionaries:



d = {}

d["Len"] = 100.0
d["Stephen"] = 78.0


for key, value in d.items():
   print("Score %s is %s " % (key,value))









if "Len" in d:
   print("Found!")
```

C++

```
// Voorbeeld met map:
#include <map>
using namespace std;

map<string, float> d; // key en value type nodig

d["Len"] = 100.0;
d["Stephen"] = 78.0;

// itereer over paren van <key,value> van begin() tot
// end() (is 1 positie na laatste)
for(map<string, float>::iterator it = d.begin();
    it != d.end(); it++){
   // it verwijst naar pair<string,float> object
   // met members first en second
   cout << "Score " << it->first << " is "
                    << it->second << endl;
}

if(d.find("Len") != d.end()){
   cout << "Found!" << endl;
}
```

Part 4: If

Python

```python
# If example
x = 10
y = 11
if x==10 and y!=12:
   print(True)


# If-else example
if x==9 or y==12:
   print("hier kom ik niet")
else:
   print("Juist!")


# Zoek element in matrix
q = [[1,2,3], [4,5,6]]


zoek = 3
for rij in range(0,len(q)):
   for kol in range(0,len(q[rij])):
       if q[rij][kol] == zoek:
          print("Element %d gevonden op rij %d " \
                "en kolom %d." % (zoek, rij, kol))

# print Element 3 gevonden op rij 0 en kolom 2.
```

C++

```cpp
// If example
int x = 10;
int y = 11;
if( x==10 && y!=12){
   cout << true << endl;       //print 1, dan nieuwe lijn
}

// If-else example
if(x==9 || y==12)
   cout << "hier kom ik niet" << endl;
else
   cout << "Juist!" << endl;


// Zoek element in matrix
vector<vector<int> > q = {{1,2,3},{4,5,6}};
                          //spatie bij '>>' verplicht
int zoek = 3;
for(int rij=0; rij < q.size(); rij++){
   for(int kol=0; kol < q[rij].size(); kol++){
       if(q[rij][kol] == zoek){
          cout << "Element " << zoek
               << " gevonden op rij " << rij
               << " en kolom " << kol << endl;
       }
   }
}
//print Element 3 gevonden op rij 1 en kolom 3.
```

Part 5: For & While

Python

```python
# while loop: print 0,1,3,4,5
i = 0
while i < 6:        # stop als i == 6
    print(i)        # indentatie verplicht!
    i=i+1           # als ontbreekt: oneindige loop

print(i)            # print 6


# while loop: gemiddelde lijst
lijstje = [5.0, 9.5, 5.5]
i = 0
som = 0.0
while i < len(lijstje):        # stop als i == 3
    som += lijstje[i]
    i=i+1

print("Gemiddelde is %.3f" % (som/len(lijstje)))

# while loop: nieuwe lijst van kwadraten
i = 0
kwadraten_lijst = []
while i < len(lijstje): # stop als i == len(lijstje)
    kwadraat = lijstje[i]*lijstje[i]
    kwadraten_lijst.append(kwadraat) # voeg toe
    i=i+1



print(kwadraten_lijst)       # [25.0, 90.25, 30.25]



# for loop: print 0,1,2,3,4,5
for i in range(0,6,1):
    print(i)
```

C++

```cpp
// while loop: print 0,1,3,4,5
int i = 0;
while(i < 6){             //stopt als i == 6
    cout << i;
    i=i+1;                //als ontbreekt: oneindige loop
}
cout << i << endl;        //print 6


// while loop: gemiddelde lijst
vector<double> lijstje = {5.0, 9.5, 5.5};
int i = 0;
double som = 0.0;
while(i < lijstje.size()){   // stop als i == 3
    som += lijstje[i];
    i=i+1;}

cout << "Gemiddelde is " << som/lijstje.size() << endl;

// while loop: nieuwe lijst van kwadraten
i = 0;
vector<double> kwadraten_lijst;
while(i < lijstje.size()){//stop als i == lijstje.size()
    double kwadraat = lijstje[i]*lijstje[i];
    kwadraten_lijst.push_back(kwadraat); // voeg toe
    i=i+1;
}
// range for loop
// opm.: gewone cout toont enkel primitieve waarden
for(double i: kwadraten_lijst)
    cout << i << " ";
}

// for loop, print 0,1,2,3,4,5
for(int i=0; i<6; i++){
    cout << i;
}
```

Part 6: Functions

Python

```python
# Eerste voorbeeld: fahrenheit2celsius


''' Function with name fahrenheit2celsius and one
input parameter 'f'. The function returns a single
value '''
def fahrenheit2celsius(f):
    c =  5.0/9.0 * (f - 32)
    return c

''' Function with name printResultsF2C and two input
parameters 'f' and 'c'. This function does not
return a value '''
def printResultsF2C(f,c):
  print("Fahrenheit: %-6.3f Celsius: %-6.3f" %
(f,c))


''' Call function with name fahrenheit2celsius
f parameter gets value of argument f1 with value
40.0, and function returns value of c, which is
stored as c1 in caller program.'''
f1 = 40.0
c1 = fahrenheit2celsius(f1)
printResultsF2C(f1,c1)




# Tweede voorbeeld: bereken gemiddelde van elke rij
# in matrix.
mydata = [[1,11,5], [7,14,10], [10,100,20]]


''' Met functies: Function with name 'gemiddelde'. A
is the parameter of type list.  Function returns a
float value.'''
def gemiddelde(a):
    gemiddelde = 0.0
    for val in a:
        gemiddelde += val
    gemiddelde = gemiddelde/float(len(a))
    return gemiddelde;









# Oproep functie gemiddelde.
for i in range(0,len(mydata)):
    rij = mydata [i]
    gemiddelden[i] = gemiddelde(rij)

print("Gemiddelden zijn %s" % gemiddelden)
```

C++

```cpp
// Eerste voorbeeld: fahrenheit2celsius
#include <iomanip>            //voor setprecision

/* Function with name  fahrenheit2celsius and one input
parameter 'f'. The function returns a single value.
*/
double fahrenheit2celsius(double f){
    double c =  5.0/9.0 * (f - 32);
    return c; }

/* Function with name printResultsF2C and two input
parameters 'f' and 'c'. This function does not return a
value.*/
void printResultsF2C(double f, double c){
    cout << "Fahrenheit: " << setprecision(3) << f
         << " Celsius: " << setprecision(3) << c << endl;
}

/* Call function with name fahrenheit2celsius
f parameter gets value of argument f1 with value 40.0,
and function returns value of c, which is stored as c1
in caller program.*/
double f1 = 40.0;
double c1 = fahrenheit2celsius(f1);
printResultsF2C(f1,c1);




// Tweede voorbeeld: bereken gemiddelde van elke rij in
// matrix.
vector<vector<int>> mydata = {{1,11,5}, {7,14,10},
                              {10,100,20}};

/* Met functies: Function with name 'gemiddelde'. A is
the parameter of type vector. Function returns a float
value.*/
double gemiddelde (vector<int> a){
    double gemiddelde = 0.0;
    for(int val: a)
        gemiddelde += val;
    gemiddelde = gemiddelde/a.size();
    return gemiddelde;
}

// Hulp functie om vector af te drukken
string vecToStr(vector<double> row){
     string s;
     for(double d: row){
         s += to_string(d) + " "; // convertie
     }
     return s;
}

// Oproep functie gemiddelde and vecToStr.
for(int i=0; i<mydata.size(); i++){
     vector<int> rij = mydata[i];
     gemiddelden[i]= gemiddelde(rij);
}
cout << "Gemiddelden zijn " << vecToStr(gemiddelden)
      << endl;
```

Part 7: Classes

Python

```python
# Voorbeeld met klasse:



class Person:



  def __init__(self, name, age): // constructor
    self.name = name  // fields
    self.age = age


  def myfunc(self):   // methode
    print("Hello my name is " + self.name)







if __name__ == "__main__":
   p1 = Person("John", 36)
   p1.myfunc()
```

C++

```cpp
// Voorbeeld met klasse:
#include <iostream>    //voor cout
#include <string>
using namespace std;

class Person{
public:   // variabelen en methodes zichtbaar buiten
          // klasse Person

  Person(string name, int age){ // constructor
      this->name = name;          // zie private sectie
      this->age = age;
  }

  void myfunc(){      // methode
    cout << "Hello my name is " << this->name << endl;
  }

private:  // variabelen en methodes niet zichtbaar
          // buiten klasse code Person
  string name; // fields
  int age;
};

int main(){
    Person p1("John", 36);
    p1.myfunc();
}
```

Part 8: Nested List/Vector

Python

```python
# Maak matrix (met nested lists)
q = [[1,2,3], [4,5,6], [7, 8, 9]]
first_first = q[0][0]       # 1
row_two = q[1]              #[4,5,6]
last_last = q[-1][-1]       # 9

# Maak matrix met variabele grootte
rijen = []
for i in range(0, 10):
   rij = []
   for j in range(0, 10):
      rij.append(0.0)
   rijen.append(rij)



# Nested loop: print elke rij
for rij in range(0,len(q)): # exclusive 3, row=0,1,2
   print("Row %d:" % rij)
   for kol in range(0,len(rij)):
       print(q[rij][kol])

# print row 0: 1,2,3 …

# som van alle elementen in matrix:
som = 0
for rij in range(0,len(q)):           #range(0,3)
   for kol in range(0, len(q[rij])):
       som = som + q[rij][kol]


# som is q[0][0] + q[0][1] + q[0][1] + … = 45
```

C++

```cpp
// Maak matrix (met nested vectoren)
vector<vector<int>_> q = {{1,2,3},{4,5,6},{7,8,9}};
int first_first = q[0][0];     // 1
vector<int> row_two = q[1];    // [4,5,6]
int last_last = q[q.size()-1][q.size()-1];  // 9


// Maak matrix met variabele grootte
vector<vector<double>_> rijen;
for(int i=0; i<10; i++){
   vector<double> rij;
   for(int j=0; j<10; j++){
       rij.push_back(0.0);
   }
   rijen.push_back(rij);
}

// Nested loop: print elk rij
for(int rij =0; rij<q.size(); rij++){
   cout << "Row " << rij << ":" << endl;
   for(int kol =0; kol<q[rij].size(); kol++){
       cout << q[rij][kol] << " ";
   }
} //print row 0: 1,2,3 ….

// som van alle elementen in matrix:
int som = 0;
for(int rij=0; rij<q.size();rij++){
    for(int kol=0; kol<q[rij].size(); kol++){
       som = som + q[rij][kol];
    }
}
// som is 45
```