Use Python in the Information Technology Practice Exam

László Gugolya
[*] OE AMK, Székesfehérvár, Hungary
gugolya.laszlo@amk.uni-obuda.hu

Abstract— **Since 2005 new graduation system in Hungary. In this context, the advanced level practical exam should also be subject to programming. Here you can choose between the candidates of the programming languages. The optional languages later (2012) was added to the Python language. This language is widely used for educational purposes in the US universities. The language is spreading in Hungary, that look good to the baccalaureate exams students at elementary and secondary level elevated nearly 10% of this. More and more people are chosen from among the teachers in the classroom and students ' exams.**

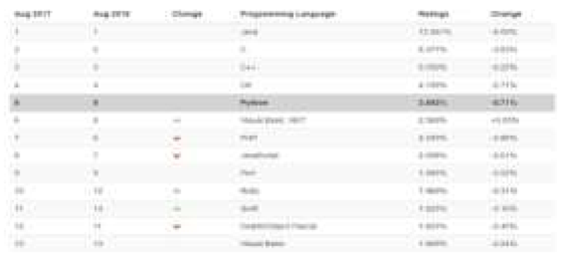
This trend has been placed under investigation, following which the other options for the language in the language compared to the baccalaureate curriculum. What are the advantages and disadvantages are to be expected in the course of teaching and learning.

I. Introduction

In the current maturity system, the candidate can choose from several programming languages. Thus, the question arises as to which language to choose the candidate and the teacher preparing the exam. There are several aspects to consider here. In the case of individual preparation, the candidate intends to continue to study as an important aspect. If you are going to learn Visual Basic in the higher education institution (eg economic informatics), then it is best to choose this option. This is not possible in a school or group environment. In this case, other considerations arise, such as the prior knowledge of the teacher and the usable time frame. Over the last few years, the number of hours spent on computing and within programming has decreased. Consequently, teachers are seeking ways to search for opportunities.

At present (2017), maturity exams can be selected from Pascal, C ++, C #, Visual Basic, Java, Python languages [1]. According to the programming language topology (TIOBE index) [2], this is a list of trends, from which Pascal hangs a bit from the line. This programming language has historically played a significant role in the Hungarian computer science education and is currently being implemented. So it can be justified to include it on the list.

I have been preparing for graduation for many years, but so far Python has not been educated. Several colleagues have a positive experience of using and teaching the language. So I came to see the time to look more closely at the potential. If I get caught, I tried to read on the Internet or read it in the literature [4].



[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]
[illegible]	[illegible]		Python	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]

Figure 1. TIOBE lista, 2017 augusztus

A. Dating

After the first chat, reading the Internet, you can place the language you want to know. Python is a general purpose, high level programming language. When designing the language, the readability and facilitation of programming work were emphasized.

Python supports functional, object-oriented, imperative, and procedural programming paradigms. It uses dynamic types and automatic memory management. The Python interpreter language, ie the source and object code is not separated.[3]

We can use it in a variety of areas: for web applications, desktop applications, game development, but many systems use it as a snap language (SPSS, PostreSQL).

There are several tools for programming. You can choose between IDLE and WinPython in maturity exams. For the learning process, we can use any of our favorite text editor, as it is widely supported because of the prevalence of language. Larger systems also support this language, so they can be used in Visual Studio, Eclipse, Netbeans, as well. There are some systems that prefer this language. Such are the many popular PyCharms and NINJA IDEs. For the first steps, I used Komodo Edit, then I asked for a 1-year educational license for PyCharm and started learning about language.

II. First Steps

A. Language elements

When I get acquainted with the language elements, I use the 2017 code of the general exam level exam. You can download the text of the task and its source file from http: //oktatas.hu. The official solution was released this year in C #. During the initial steps, I tried to use simple solutions. An important aspect was the ability to teach in learning.

Using the Python Console is a useful tool at the beginning of getting started. Here, you can see the result of our instructions, and any outputs. This is very useful during the first step of the educational process.

The first significant difference is the significance of formatting compared to the other languages used in the exam. Missing the instruction block with the characters. This can be done by formatting. This in the first place is an advantage, as educational experiences show that it is natural for students. In the case of other languages, the use of code templates has less problems in this field.

```
for m in tesztek:
    if beKod == m["kod"]:
        print(m["megoldas"])
        keresett = m
```

The counting cycle shown in the previous example forms a statement block. The statement block of the conditional statement consists of the "print" and the assignment instructions.

The assignment was the usual marking. It is possible to multiply it. This can be seen in the first line. Multiple Assignments can be used in another form. this is equivalent to a = 0; b = 1. This instruction allows you to exchange values for variables. This is shown in the third row.

```
a=b=c=2
a, b = 0, 1
a, b = b, a
```

Conditional statements do not include a multiple conditional statement. This if ... elif ... must be solved. When using the terms, it is possible to use the relationship of mathematics classes. Example of solving problem 6.

```
if k <= 5:
    pontszam += 3
elif 6 <= k <= 10:
    pontszam += 4
elif 11 <= k <= 13:
    pontszam += 5
elif k == 14:
    pontszam += 6
```

The language knows the conditional term. Its shape differs from the usual C-like language.

```
kisebb = a if a<b else b
```

For cycles, the back tester is not among the language elements. This can be replaced by the first time with the (while) test.

```
while True:
    utasítások
    if <kilépési feltétel>:
        break
    (további utasítások)
```

Here is another language for the break and continue statements. They only apply to one cycle.

The "for" cycle can be used extensively. Its operation is fundamentally different from the usual. It does not go through a series of numbers, but it does enter something. That is, it is not a "counting" cycle, but a "crawling" cycle. Here is no cycle variable in other languages. This is more like a foreach in C #. We can go through all the elements of a multitude (eg a list). For example, for task 5.

```
dbJo = 0
for m in tesztek:
    if helyes[s] == m["megoldas"][s]:
        dbJo += 1
```

You can also use cycles in numeric order using a function. The range () function generates an interval as a return value, and the resulting list goes through the for cycle. This way we can produce a cycle close to a traditional one. This could be used to map the solutions stored in the string by character in task 6. Here, since 14 questions were included in the test, the range () function produces a 0-13 interval.

```
for k in range(0,len(helyes)):
    if helyes[k] == m["megoldas"][k]:
        if k <= 5:
            pontszam += 3
        elif 6 <= k <= 10:
            pontszam += 4
        elif 11 <= k <= 13:
            pontszam += 5
        elif k == 14:
            pontszam += 6
```

Repetitive structures may have another branch. This is different from the other programming language that can be used for the exam. This branch will be executed if the cycle has run through the list (for case) or if the condition is fake (while). It will not be executed if the cycle is interrupted by the break command.

```
while <feltétel>:
    utasítások
else:
    utasítások
```

Many people like to solve each task individually. This will make the solution more understandable. To do this you need to know how to use the functions. The function is specified as follows.

```
def feladat2():
    print("2. feladat: ")
    print(" A vetélkedőn {0} versenyző
indult.".format(len(tesztek)))
```

You can specify parameters as usual, return the value to the return. Here's an interesting opportunity to see, this is multiple value reproduction. In the example below, 6 and 9 are printed.

```
def fuggveny(x):
    return x*2, x*3
a,b = fuggveny(3)
```

```
print (a," ",b)
```

It is also possible to enter local functions similarly to the Pascal language.

```
def fgv():
    def alfgv():
        print("alfgv vagyok")
    print("fgv vagyok")
    alfgv()
```

The data structure like arrays is multi way in Python: list, tuple, dictionary.

The most versatile composite data type of Python is a list (list) that can be entered as comma separated values in square brackets. The elements in the list do not have to be of the same type.

```
gyumi=["alma",260, True, 12.5]
gyumi2=[["alma","körte","meggy"],260]
```

The second line shows that the list item can be a list. You can refer to the element of that list by block from 0. It is also possible to refer to several elements of the list (slices, parts), for example:

```
gyumi[1:3]->[260, True]
```

It is also possible to access the list from the back, so negative indices are used. For example, the last element can be referred to as a gyumi[-1].

To manage lists, you can use append, extend, insert, remove, pop, index, count, sort, reverse. These can be used as stack or row data structures. This is useful for problem solving.

"Two-dimensional array" is used in maturity exams. We can do this with a list in the list. This is like a one-dimensional array implemented in one-dimensional array in other languages.

```
lista=[[2,3],[5,6],[8,9]]
s = ""
for x in lista:
    for y in x:
        s += " " + str(y)
    s += "\n"
print(s)
```

Listing is facilitated by so-called "list mapping". You can then perform a particular action on all the items on the list, and then create a new list. You can view it quickly on the console.

```
>>> lista=[1,2,3,4]
>>> lista = [elem*2 for elem in lista]
>>> lista
[2, 4, 6, 8]
```

Like a list, a structured data structure for storing different objects is tuble. Contrary to the list, the elements can not be modified here. This may be useful if you have to work with fixed elements, such as the names of days and months.

```
napok = ('hétfő','kedd','szerda','csütö
rtök','péntek','szombat','vasárnap')
```

Use dictionary to perform record-like data storage. You can store key-value pairs here. As an example, the 2013 "Választás" taskbar, where you can enter pairs of party abbreviations and names. Using the dictionaries with a list, we can implement traditional, structured, record-keeping data storage.

```
partok={"GYEP":"Gyümölcsevők Pártja",
        "HEP":"Húsevők Pártja",
        "TISZ":"Tejivók Szövetsége",
        "ZEP":"Zöldségevők Pártja",
        "-":"Független jelöltek"}
```

The String object can not be modified. Its use is the same as in the lists, that is, a list of roundabouts. We have methods for implementing string actions. Of these, the use of the strip should be highlighted. After scanning (console, file), we need to remove whitespace characters. You can do this with the strip ().

Handling sets as a standalone data type can be implemented. Creating it with set (). This had to be repeatedly used in the maturity quiz of recent years. For example, in the May 2013 taskbar, to define themes.

```
temak=set()
for tema in adatok:
    temak.add(tema)
print("A temakörök:", ",".join(temak))
```

You can create a set from an existing list.

```
>>> kosar = ['alma', 'meggy', 'alma', '
cseresznye', 'meggy', 'alma']
>>> gyumi = set(kosar)
>>> gyumi
{'meggy', 'alma', 'cseresznye'}
```

The usual actions can be done on the sets: containment, embedding, deleting, engraving, union, difference, symmetric difference (in, add, pop, remove, discard, &, |, ^).

In tasks, it is often necessary to sort the stored data. We now have the option to use traditional sorting algorithms, but the language provides a way to sort the lists.

```
lista = [5, 3, 4, 1, 2]
for i in range(len(lista) - 1):
    for j in range(i + 1, len(lista)):
        if lista[i] > lista[j]:
            lista[i], lista[j] = lista[
j], lista[i]
print(lista)
lista = [5, 3, 4, 1, 2]
lista = sorted(lista)
print(lista)
lista = [5, 3, 4, 1, 2]
```

```python
lista.sort(reverse=True)
print(lista)
```

In the example, the bubble algorithm is first seen. then the shorted () function, which creates a new list that is generated as a parameter. In the third solution, sorting is done locally. Sorting is done in descending order by specifying the parameter. For a more complex list, sorting () is sorted by the first "column". If you want something else, you can set this with the key parameter.

```python
import collections
Diak = collections.namedtuple("Diak","Nev,Szulido,Magassag,Tomeg")
peldaLista=[]
peldaLista.append(Diak("Nagy Anna","2000-10-10",170,70))
peldaLista.append(Diak("Szép Éva","2012-10-10",120,30))
peldaLista.append(Diak("Nagy Attila","2011-1-21",180,68))
print(*sorted(peldaLista,key=lambda adat:adat.Tomeg),sep='\n')

peldaLista2=[]
peldaLista2.append(["Nagy Anna","2000-10-10",170,70])
peldaLista2.append(["Szép Éva","2012-10-10",120,30])
peldaLista2.append(["Nagy Attila","2011-1-21",180,68])
print(*sorted(peldaLista2,key=lambda adat:adat[3]),sep='\n')
```

In the first example, we see a solution when naming the column data stored in the list and naming it for the sort key. In the second example there is a list with a list to complete the sorting. You will then need to enter the "column" number at the key.

Reading like array:

```python
fajl=open("valaszok.txt")
adat =[]
for e in fajl.readlines():
    adat.append(e.strip().split())
print(adat)
```

Reading like record in the dictionary:

```python
adat2=[]
fajl=open("valaszok.txt")
i=1
for e in fajl.readlines()[1:-1]:
    (kod,megoldas) = e.strip().split()
    valasz = {
        "sorsz": i,
        "kod": kod,
        "megoldas": megoldas
    }
    adat2.append(valasz)
    i+=1
print(adat2)
```

In case of random number generation, we can choose from several solutions.

```python
import random
print(random.randint(1, 6))
print(random.random() * 100)
print(random.choice(['alma', 'meggy', 'cseresznye']))
print(random.randrange(0, 101, 5))
```

The autumn maturity of 2015 had to produce random coins at random, so we can easily do that on the basis of the above.

```python
import random
print("A pénzfeldobás eredménye:",random.choice("IF") )
```

In the 2016 "Zár" task sequence, a series of codes had to be produced. This is also easy to solve. With the sample method, you can create a given number of samples from a given pattern and then merge it. You can see it on a bracket.

```python
sorozat = random.sample("0123456789",5)
sorozat
['0', '3', '5', '7', '8']
print("".join(sorozat))
03578
```

B. Experience in solving task series

After the basic elements of the language I tested a complete set of tasks. I solved the last "Test Competition" task in 2017. This can be considered as a mixed task, as it is possible to use record-based thinking, and handling text data (strings) is also needed. The exact description of the task and the source file can be found at https://dari.oktatas.hu/kir/erettsegi/okev_doc/erettsegi_2017/e_inf_17maj_fl.pdf.

The task series evaluates a contestant's response to test tasks by using a text file. The tasks have been solved by means of subprograms for ease of comprehension.

The first task is to read the data. The text file has been included in a global list of data.

```python
def feladat1():
    print("1. feladat:")
    adatok=[]
    fajl=open("valaszok.txt")
    global helyes
    helyes = fajl.readline().strip()
    #tovabbi adatok beolvasasa
    i = 1
    for sor in fajl:
        sor = sor.strip()
        reszek = sor.split()
        valasz = {
            "sorsz": i,
            "kod": reszek[0],
            "megoldas": reszek[1]
        }
        adatok.append(valasz)
        i = i+1
    #print(adatok)
    fajl.close()
    return adatok
```

The second task is for the starting competitor. This can be done by using the length of the list.

```
def feladat2():
    print("2. feladat: ")
    print(" A vetélkedőn {0} versenyző indult.".format(len(tesztek)))
```

The third task is to request a competitor's details. Here is a linear search query.

```
def feladat3():
    print("3. feladat:")
    global keresett
    beKod = input("A versenyző azonosítója = ")
    for m in tesztek:
        if beKod == m["kod"] :
            print(m["megoldas"])
            keresett = m
```

In the fourth task, the correct solutions of the competitor requested in the given form have to be given. The "sought-after" contestant's solutions are per character and weigh the output into a text variable.

```
def feladat4():
    print("4. feladat")
    global helyes
    global keresett
    print("{0:s} (a helyes megoldás)".format(helyes))
    ki = ""
    for k in range(0,len(keresett["megoldas"])):
        if helyes[k]==keresett["megoldas"][k]:
            ki =  ki + "+"
        else:
            ki = ki + " "
    print("{0:s} a versenyző helyes válaszai".format(ki))
```

In the fifth task, the success of a given task solution has to be expressed in percentage. When a solution is made, a count is to be made.

```
def feladat5():
    print("5. feladat")
    beFeladatSorszama = int(input("A feladat sorszáma = "))
    dbJo = 0
    for m in tesztek:
        if helyes[beFeladatSorszama] == m["megoldas"][beFeladatSorszama]:
            dbJo += 1
    print("A feladatra {0} fő, a versenyzők {1}%-a adott helyes választ.".format(dbJo,round(dbJo/len(tesztek)*100,2)))
```

In the sixth task, the score of the competitors must be saved to an output file. Scores are given in advance for each task. Two embedded cycles should be used in the solution. One of the contestants goes the other way for each competitor's responses. You can also use elseif for the scores. The solution described illustrates the standardization in mathematics, which is different from other programming languages. The list of points7 will be useful when solving the seventh task.

```
def feladat6():
    print("6. feladat:")
    global helyes
    global pontok7
    kiFajl = open("pontok.txt","w")
    for m in tesztek:
        pontszam = 0
        for k in range(0,len(helyes)):
            if helyes[k] == m["megoldas"][k]:
                if k <= 5:
                    pontszam += 3
                if 6 <= k <= 10:
                    pontszam += 4
                if 11 <= k <= 13:
                    pontszam += 5
                if k == 14:
                    pontszam += 6
        kiFajl.write("{0} {1}\n".format(m["kod"],pontszam))
        pontok7.append([pontszam,m["kod"]])
    kiFajl.close()
```

In the seventh task, prizes must be awarded based on the scores obtained. First of all, we perform the orderly scoring according to the scores, and then the prizes are paid out to ensure that the prize winners do not lose.

```
def feladat7():
    print("7. feladat:")
    global pontok7
    rendezett = sorted(pontok7, reverse = True)
    db = 1
    i = 0
    while i<len(rendezett) and db <= 3:
        print(db, '.díj','(',rendezett[i][0], 'pont):', rendezett[i][1])
        if i + 1 < len(rendezett) and rendezett[i+1][0] != rendezett[i][0]:
            db += 1
        i += 1
```

The task was not particularly difficult. Python could be solved in 45 minutes without practice.

III. SUMMARY

Compared to the solutions of Python's own and others, there is no significant difference between the official solutions and the comparison. Regarding solutions, there is no difference in the number of rows if we omit the blank of "{}" of the official solution. It is felt that data storage is more flexible than in other more traditional languages. In addition to the scans, the use of Python was convenient (and therefore more advantageous) for sorting.

During the study and testing of the language, it became clear to me why Python was growing fast. On the basis of the many positive feedback it can be stated that it is worth trying the language and introducing it into secondary education. However, I consider it implausible that Python does not play a significant role in the Hungarian higher education. Thus, entering the higher education of the students who are going to take part can be disadvantageous.

REFERENCES

[1] https://www.tiobe.com/tiobe-index/

[2] https://www.oktatas.hu/kozneveles/erettsegi/2017oszi_vizsgaidoszak/2017osz_nyilvanos_anyagok_listaja

[3] http://nyelvek.inf.elte.hu/leirasok/Python/

[4] Mark Summerfield, “Python 3 programozás”, Kiskapu Kiadó, 2009