**Python -1**

Bug - შეცდომა- ეწოდება შეცდომას რომელიც პროგრამის მუშაობისას ან გაშვებისას წარმოიქმნა.4

პითონში გვხვდება სამი სახის შეცდომა

1. სინაქტსური შეცდომა - syntax error
2. მუშაობის პროცესის შეცდომა - Runtime error
3. ლოგიკური შეცდომა semantic error

შეცდომის ძიების პროცესს ქვია debugging

ცვლადი -

ცვლადი იწერება დაბალი ტირეს გამოყოფით, ან ყოველი მომდევნო სიტყვა იწერება დიდი ასოთი

არსებობს დარეზერვებული სიტყვები, რომლებიც არ უნდა გამოვიყენოთ ცვლადის დაწერის დროს

მაგალითად false,

ცვლადის ტიპები

-int (integer) - მთელი რიცხვი

- float - ათწილადი რიცხვები

- str (string) - ტექსტი

- bool - ლოგიკური ცვლადი, ორი მნიშვნელობით true/false

ინფორმაციის შეტანა ხდება ინფუთით და გამოტანა პრინტით

მაგალითად: user\_input= input(“Hello World”)

Print(user\_input)

Number 1= int(input(“enter the number”)) - შეიყვანს მთელ რიცხვს რადგან ინტეჯერი არის მითითებული

არითმეტიკული ოპერატორები

|  |  |  |  |
| --- | --- | --- | --- |
| სინტაქსი | აღწერა | მაგალითი | შედეგი |
| + | შეკრება | 4+2 | 6 |
| - | გამოკლება | 4-2 | 2 |
| \* | გამრავლება | 2\*3 | 6 |
| / | გაყოფა | 6/2 | 3.0 |
| // | გაყოფა მთელი | 6//4 | 1 |
| % | გაყოფა ნაშთით | 5%3 | 2 |
| \*\* | ხარისხში აყვანა | 2\*\*3 | 8 |

**Python 2- პირობითი ინსტრუქციები და მათი სინტაქსი**

მინიჭების ოპერატორები:

-= ნიშნავს დაკლებას

მაგალითად -=2 ორით დაკლება

+=მომატება, \*=გამრავლება

ლოგიკური ოპერატორები

And - და აბრუნებს თრუს თუ ორივე პირობა სწორია

or- ან აბრუნებს თრუს თუ ერთ-ერთი პირობა მაინც სწორია

not- აბრუნებს პირობით შებრუნებულ მნიშვნელობას,თუ პირობა არის თრუ დააბრუნებს ფოლს

არსებობის/წევრობის ოპერატორები

In - აბრუნებს თრუს თუ ობიექტი შეიცავს მითითებულ მნიშვნელობას

Not in - აბრუნებს თრუს თუ ობიექტი არ შეიცავს მითითებულ მნიშვნელობას

შედარების ოპერატორები

<- მეტია, >- ნაკლებია, == ტოლია, != ტოლი არ არის, >= მეტია ან ტოლია, <= ნაკლებია ან ტოლია

If number 1>number 2:

Print (number 1)

პირობის მერე იწერება ორი წერტილი, თუ გვაქვს ისეთი ლოგიკა რომელიც უნდა ამუშავედეს იფ პირობის შესრულების შემთხვევაში , ეს უნდა დაიწეროს ქვემოთ 4სფეისის გამოტოვებით, ინდენტაციით.

If statement:

Print (“number1”)

else:

print (“if else finished”) ელსის მერეც იწერება ორი წერტილი

else if - ის მაგივრად არის elif

if statement:

print()

elif statement2:

print ()

else:

print () ,

elif შეიძლება რამდენიმე ცალი იყოს

პითონში შეგვიძლია if – elif- else კონსტრუქციები ჩავდგათ ერთმანეთში, რაც ნიშნავს რომ იფში ჩადგმული იფ ამოქმებდდება მხოლოდ პირველი იფის პირობის შესრულების შემთხვევაში, ჩადგმული იფი კი ისევ ინდენტაციის დაცვით უნდა დავწეროთ, ერთი იფის შემდეგ მეორე იფი, მერე მესამე, ყოველ ჯერზე გადავა სფეისებით მარჯვნივ , და ყოველი იფის ქვეშ თავისი ელსე დაიწერება.

**Python 3 - ციკლები, range**

**იტერატორი -** არის ობიექტი, რომელიც შეიცავს მნიშვნელობის თვლად რაოდენოას.. ტქნიკურად პითონში იტერატორი არის ობიექტი, რომელიც ახორციელებს იტერატორ პროტოკოლს რომელიც შედგება მეთოდებისაგან \_iter\_() და \_next\_()

იტერატორი გარდაქმნის ობიექტად ციკლებს , ერეის, მონაცემებს, ხოლო ნექსთი კითხულობს შემდეგ ამ ობიექტს. ნექსთი კითხულობს მხოლოთ თითო ობიექტს, სიმბოლოს

მაგალითი: text=”hello world”

Iterator\_text=iter(text)

Print(next(iterator\_text)) -iterator ერთ კონკრეტულ ინტეჯერზე(რიცხვზე) არ გამოიყენება

მაგრამ თუ მასივია ინტეჯერებისგან შემდგარი მაშინ გამოიყენება.

იტერაცია არის ერთიდაიგივე ბლოკის რამდენჯერმე გამოყენება

While and for - უსასრულო და სასრული. მოქმედებს მანამ სანამ მისთვის გადაცემული პირობა არის თრუ

While გადაეცემა ისეთი პირობა რომელსაც აქვს თრუ ან ფოლს მნიშვნელობა, იწერება ბოლოში ორი წერტილი და მერე ხდება ბლოკის აღწერა ,რა უნდა მოხდეს თუ პირობა სრულდება.

აქაც ინდენტაციით იწერება. და ბოლოში ელსი , მაგრამ ყოველთვის არ იწერება. თუ ბლოკი სრულდება სწორედ ბოლომდე მაშინ ხორციელდება ელსი.

Break and Continue

While (condition)

Statement

Statement

Continue - თუ პირობა სრულდება მაშინ ახტება მის შემდეგ ლოგიკას , და თავიდან იწყებს

Break- ის დროს წყდება ყველა ბლოკს, ასრულებს ციკლს და აღარ ახორციელებს მის მერე დაწერილ ბლოკს

Statement

Statement

**Number=10**

**While number>0:**

**Print (number) უსასრულოდ ატრიალებს 10 იანს**

**While number >0:**

**Print (number)**

**If number == 2**

**break**

**Number -=1**

**Else:**

**Print (“number is 0”)**

**თითო დატრიალებაზე დააკლებს 1 -ს სანამ განოლდება. განოლების მერე გაჩერდება**

**For – is used for iterating over a sequence and iterables like range, list. Or a string.**

**იქმნება დროებითი ცვლადი აქ i შემდეგ in - ს და სად აქ რეინჯი**

**For i in range (5):**

**Statement1**

**Statement2**

**Else:**

**Statement**

**Range - აბრუნებს მხოლოდ ციფრებს**

1. **მხოლოდ საბოლოო მნიშვნელობა - range (10) გამოვა ათამდე**
2. **საწყისი და საბოლოო მნიშვნელობა range (5,10) ხუთი გამოვა , ათამდე**
3. **საწყისი. საბოლოო და ბიჯი- range (5,10, 2) ხუთი გამოვა, და 7 . ხუთიდან მომდევნო 2**

**For i in range (5):**

**Print (i) დაბეჭდავს 0 დან 4 ის ჩათვლით**

**Range შეგვიძლია უკუთვლით წამოვიდეთ, ამ დროს პირველ რიცხვის ვუთითებთ დიდს და მერე დაბალი რიცხვი**

**Range (10, 5,-1) დაიბეჭდება 10.9.8 . 7.6,**

**Text=”hello world”**

**For letter in text:**

**Print(letter) ვერტიკალურად გამოიტანს სიტყვებს**

print("Item1", end=" ") ბეჭდავს ჰორიზონტალურად

**Random - ფუნქციები**

**გააჩნია რამდენიმე მეთოდი**

1. **Random - აბრუნებს შემთხვევით რიცხვებს 0დან 1მდე**
2. **Randint - აბრუნებს შემთხვევით მთელ რიცხვს მითითებულ შუალედში, იგულისხმება ბოლო რიცხვი**
3. **Randrange- აბრუნებს შემთხვევით მთელ რიცხვს მითითენი; შია;ედშო არ ოჰი;ოსხ,ენა**

**ბოლო რიცხვი**

**რენდომის დროს უნდა დავაიმპორტოთ შემდეგნაირად**

**@import: random**

**Random\_number= random.random ()**

**Random\_number= random.randint (5,10) 5 დან 10 ს ჩათვლით**

**from random import: randint**

**Random\_number= randint (5,10) 5 დან 10 ს ჩათვლით**

**Python 4 – strings, სტრიქონები**

**String -** სტრინგი არის მონაცემთა ტიპი. რომელიც წარმოადგენს სიმბოლოთა თანმიმდევრობას და მონაცემთა ეს ტიპი არის უცვლელი (immutable)

String ს გააჩნია სიგრძე რომლიც შეგვიძლია გავიგოთ len() ფუნქციის გამოყენებით

სტრინგის მეთოდები

|  |  |
| --- | --- |
| **Capitalize ()** | **აკონვერტირებს მაღალ რეგისტრირებს მხოლოდ პირველ ასოს** |
| **Lower ()** | **აკონვერტირებს ყველა სიმბოლოს დაბალ რეგისტრირში** |
| **Upper ()** | **აკონვერტირებს ყველა სიმბოლოს მაღალ რეგისტრირში** |
| **Count ()** | **აბრუნებს რაოდენობას, რამდენჯერაა ტექსტში გადაცემული სიმბოლო / სიტყვა** |
| **Index ()** | **სტრინგში ეძებს გადაცემულ სიმბოლოს და გვიბრუნებს მის ინდექსს** |
| **Split ()** | **ყოფს სტრინგს შესაბამისი გამყოფით** |
| **Strip ()** | **აშორებს ცარიელ სიმბოლოებს** |
| **Format ()** | **გვეხმარება სტრინგში მნიშვნელობის შესაცვლელად** |

**სტრინგზე მოქმედებები**

|  |  |  |
| --- | --- | --- |
| **+** | **სტრიქონებს უმატებს ერთმანეთს** |  |
| **\*** | **გაამრვალებს სტრიქონს ნამრავლის რიცხვის მიხედვით** |  |
| **[]** | **დააბრუნებს სიმბოლოს შესაბამისი ინდექსით** | **a[1]=”e”** |
| **[:]** | **დააბრუნებს სიმბოლოებს მოცემული ინდექსების დიაპაზონში** | **a{1:4}=”ell”** |
| **In** | **დააბრუნებს თრუს თუ სიმბოლო არის სტრიქონში** |  |
| **Not in** | **დააბრუნებს თრუს თუ სიმბოლო არ არის სტრიქონში** |  |

**Full\_string=”Hello World” - სლაისინგი**

**Print (full\_string [1])დაბეჭდავს პირველ სიმბოლოს**

**Print (full\_string [-1])დაბეჭდავს ბოლო სიმბოლოს**

**Print (full\_string [1:10])დაბეჭდავს პირველიდან მეცხრეს ჩათვლით**

**Print (full\_string [1:])დაბეჭდავს პირველიდან ბოლომდე**

**Print (full\_string [:10])დაბეჭდავს პირველიდან მეათე სიმბოლომდე**

**Print (full\_string [::-1])დაბეჭდავს სიტყვას შებრუნებულად**

**Format – ის გამოყენება**

**Name=John**

**Age =30**

**Full \_string= “hello I am {} . I am {} years old” .format (name, age)**

**Full \_string= “hello I am {1} . I am {0} years old” .format (name, age)**

**Full \_string=f “hello I am {name} . I am {age} years old”**

**Encode (ბიტულ ფორმატში გადაცემა) and decode (გადმოტანა კოდიდან სტრინგად)**

**Txt=”hello”**

**Print (type(txt)) – ტიპი იქნება სტრინგი**

**Txt=txt.encode(“utf-8”)**

**Txt=txt.decode(“utf-8”)**

**ქსელში როდესაც გადავცემთ მაშინ უნდა გამოვიყენოთ ინქოუდინგი**

**Ascii\_char =ord (“A”), print (ascii\_char) დაბეჭდავს ციფრს რომელზეც არის A, asci ცხრილში**

**Regular Expression (RegEx) – არის სიმბოლოების თანმიმდევრობა, რომელიც განსაზღვრავს ძიების ნიმუშს, ფაილში უნდა დაიმპორტდეს import re**

**Method- ები**

**Findall () -აბრუნებს მთლიან ლისტს, რომელიც შეიცავს ყველა დამთვხვევას**

**Search () - აბრუნებს ობიექტს, თუ დამთხვევა იპოვა სტრიქონში**

**Split ()- აბრუნებს ლისტს, სადაც სტრიქონი იყოფა თითოეულ შესატყვისზე**

**Sub () - ცვლის შესატყვისებს სტრიქონში**

**Pattern - ქვია რეჯექსის სინტაქსს**

**W3school – python regex**

**მაგალითი: მოვძებნოთ ხმოვნები სტრინგში**

**Import re**

**Text=”Python is the best programming language”**

**Pattern=r”[aeiou]” [] – range**

**Matches= re.findall (pattern, text)**

**Print (matches)**

**Pattern= r”b.\*t” – ეს ნიშნავს გამრავლება რომ ნებისმიერი სიტყვა მოძებნოს რომელიც იწყება ბ ზე და მთავრდება თ ზე და შუაში რამდენი ასოც არ უნდა იყოს დაიბეჭდება ყველა სიტყვა**

**დამხმარე საიტი რომელიც გვეხმარება რეჯექსის დაბეჭდვაში - regex101.com**

**Python 5- სიები, list**

**List - შეიძლება ინახავდეს სხვადასხვა ინფორმაციას, არის ცვალებადი, მონაცემები შეიძლება იყოს ბულიანი, სტრინგი და ინტეჯერი. ლისტი ინახება ცვლადში. ცვლადი მონაცემთა ტიპია.**

**სტრინგი არ არის ცვლადი. იმისათვის რომ შევცვლაოთ უნდა გამოვიძახოთ ჯერ მეთოდი და მერე ცალკე ცვლადში შეინახო შეცვლილი სტრინგი. ხოლო ლისტზე პირდაპირ შეიძლება**

**მეთოდი გამოვიძახოთ და შევიტანოთ ცვლილება.**

**ლისტის შექმნა**

**Empty\_lits1= [] ან empty\_list2=list() – list ფუნქციის გამოყენებით**

**Integer\_list= {1,2,3,]**

**Mixed list [1,2, “hello”, 3,4, True]**

**Nested List [[1,2,3], [4,5,6]]**

**Clear () - ასუფთავებს list1.clear()**

**List1 = [] - ესეც ასუფთავებს მაგრამ თან პარალელურად ახალ ლისტს ქმნის**

**მაგალითი**

**List\_1 = [“Tbilisi”, “London”, 1, 2 ,3 , True}**

**List\_2 = List\_1**

**List\_1=clear () აქ წაშლის ორივე ლისტს რადგან აიდი და ლისტის ადგილიც ერთი და იგივეა**

**Print= (id(list\_1))**

**List\_1 = [“Tbilisi”, “London”, 1, 2 ,3 , True}**

**List\_2 = List\_1.copy()**

**აქ ქოფის დროს შეიქმნება იგივე ლისტი მაგრამ სხვა აიდით და ადგილით, ამიტომ წაშლის შემთხვევაში ერთი ლისტ 1 წაიშლება და ლისტ ორი დარჩება უცვლელი**

**Append - ამატებს ბოლოში ელემენტს ლისტში, push ივით არის**

**List\_1.append (“append element”)**

**Insert- ამატებს ელემენტს და ატრიბუტად გადაეცემა ინდექსი სად ამატებს და ობიექტი რასაც ამატებს list\_1.insert (2, “New York”)- მეორე ინდექსზე დაამატებს ნიუიორკს**

**Print (List\_1.index (“New York”)) დაბეჭდავს რომელ ინდექსზეა ნიუიორკი**

**წაშლა**

**Pop**

**Remove - უნდა გადაეცეს მნიშვნელობა რაც იშლებან list\_1.remove (3) რასაც ნახავს პირველ სამიანს იმას წაშლის, თუ მეორეჯერ დავწერთ მერე ისევ წაშლის. Remove - ის დროს იკარგება წაშლილი ინფორმაცია**

**List\_1.count(3)- ქაუნთის მეშვეობით დავითვლით რამდენჯერ არის ეს ელემენტი ლისტში**

**მას მერე რომ გავიგებთ რამდენჯერ არის for ის მეშვეობით დავატრიალებთ რიმუვს იმდენჯერ რამდენჯერაც არის და წაშლის**

**Count\_three=list\_1.count(3)**

**For i in range (count\_three)**

**List\_1.remove(3)**

**Print (list\_1)**

**Pop- შლის ბოლო ელემენტს, მაგრამ წაშლილ ელემენტს არ კარგავს და აბრუნებს /ინახავს**

**შეგვიძლია წაშლილი ელემენტები ცალკე ცვლადში შევინახოთ**

**მაგალითი:**

**Popped\_items=list\_1.pop () – პოპს თუ მივუთითებ ატრიბუტად ინდექსის ნომერს იმას წაშლის თუ არა მაშინ ბოლოს**

**Print (popped\_items) - აქ დაბეჭდავს იმ ელემენტებს რაც წაიშალა**

**Print (list\_1) – აქ დაბეჭდავს განახლებულ ლისტს საიდანაც ამოშლილია**

**Reverse - მეთოდი**

**List\_1.reverse() - აბრუნებს თანმიმდევრობას**

**Sort - სორტირება**

**List\_1.sort() თუ გვაქვს ინტეჯერები უფრო მართებულად აკეთებს რადგან მეტობით და ნაკლებობით აკეთებს სორტირებას ალაგებს ზრდადობით , რადგან დეფოლტად არის ფოლსი**

**ხოლო თუ მინდა კლებადობით დაალაგოს მაშინ უნდა დავწეროთ**

**List\_1.sort(reverse=True)**

**სტრინგის სორტირებაც შესაძლებელია , თუ მხოლოდ სტრინგული მნიშვნეულობები ეწერაბე ლისტში, სორტირების დროს გადაყავს უნიკოდში და კოდის ციფრული მნიშვნელობებით ასორტირებს ანბანის მიხედვით**

**Cities=[“Tbilisi”, “London”, “Washington”)**

**Cities.sort()**

**Comprehensions - მეშვეობით ხდება ინფორმაციის გადატანა ლისტში, გამოიყენება ფორი განსხვავებულად, filter**

Integer\_list1=[i for in range (10)]

Print (f”first list: {integer\_list1}”) ორივე დაბეჭდავს 10 მდე რიცხვებს

Integer\_list2=[]

For i in range(10):

Integer\_list2.append(i)

Print (f” second list: {integer\_list2}”)

Integer\_list1=[i for in range (0,10,2)] აქ გამოიტანს 0დან ლუწ რეიცხვებს 9 მდე, რადგან 0 იწყებს 10 მდე გამოაქვს 2 ბიჯის გამოტოვებით

ასევე, შეგვიძლია if- ის დამატება. მაგალითად მინდა დაბეჭდოს მხოლოდ ლუწი რიცხვები

List1=[5,7,9,10,11,12,14]

Even\_list=[i for i in list1 if i%2==0]

Print (even\_list)

იგივეა რაც

Even\_list\_2=[]

For i in list1:

if i%2==0;

even\_list2.append(i)

**matrix კვადრატული მატრიცა**

**ვ**