FUNKCIONALNO PROGRAMIRANJE

Oznaka predmeta: FPR

Predavanje broj: 06

Nastavna jedinica: PYTHON,

Nastavne teme:

Preklapanje operatora. Dekoratori. Skrivanje podataka. Regularni izrazi (madifikatori, zagrade, kvantifikatori, metakarakteri, match, search, sub). CGI (HTTP, varijable okruženja, get, post, forme, upload).

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

Steven Lott: "Functional Python Programming", Packt Publishing, 2015.

Python, preklapanje operatora

- Neke metode tipa `__naziv__`

  `__init__ ( self [,args...] )`,

  konstruktor (sa opcionalnim argumentima), kreira instancu klase

  primer: `obj = className(args)`

  `__del__( self )`,

  destruktor, briše objekat,

  primer: `del obj`

  `__repr__( self )`,

  string reprezentacija (sadrži je __str__),

  primer. : `repr(obj)`

  `__str__( self )`,

  printabilna string reprezentacija,

  primer: `str(obj)`

  `__cmp__ ( self, x )`,

  poređenje objekta,

  primer: `cmp(obj,x)`

Python: str, repr

- Objašnjenje str i repr:

```
>>> import datetime
>>> today = datetime.datetime.now()
>>> str(today)
'2012-03-14 09:21:58.130922'
>>> repr(today)
'datetime.datetime(2012, 3, 14, 9, 21, 58, 130922)'

>>> class Sic:
...   def __repr__(self): return 'foo'
...
>>> print str(Sic())
foo
>>> print repr(Sic())
foo
>>> class Sic:
...   def __str__(self): return 'foo'
...
>>> print str(Sic())
foo
>>> print repr(Sic())
<__main__.Sic object at 0x2617f0>
>>>
```

Python, preklapanje operatora

Preklapanje operatora

- Neka je kreirana klasa (2D) Vector koja prikazuje dvodimenzionalni vektor.
- Ideja je da se implementira metoda __add__.
- Može se definisati __add__ metoda u klasi koji će da obavlja sabiranje 2D vektora, tako da se može jednostavno pisati umesto __add__ samo znak +.

```
class Vector:
   def __init__(self, a, b):
      self.a = a
      self.b = b
   def __str__(self):
      return 'Vector (%d, %d)' % (self.a, self.b)
   def __add__(self,other):
      return Vector(self.a + other.a, self.b + other.b)
v1 = Vector(2,10)
v2 = Vector(5,-2)
print (v1 + v2)
```

```
Vector(7,8)
```

Python: skrivanje podataka

Skrivanje podatka

- Atributi objekta mogu ili ne mogu biti vidljivi izvan definicije klase
- Ako se nazivu atributa doda prefiks dve donje crte, takav atribut postaje direktno nedostupan izvan klase.
  - Sa jednom donjom crtom nedostupan je u smislu poštovanja dogovora (konvencije, *weak "internal use" indicator*).
    Ako bi uradili

    ```
    from M import *
    ```

    ne bi bili importovani objekti iz M čije ime počinje sa jednom donjom crtom.

```
class JustCounter:
    __secretCount = 0
    def count(self):
        self.__secretCount += 1
        print (self.__secretCount)

counter = JustCounter()
counter.count()
counter.count()
print (counter.__secretCount)
```

```
1
2
Traceback (most recent call last):
   File "test.py", line 12, in
<module>
      print counter.__secretCount
AttributeError: JustCounter instance
has no attribute '__secretCount'
```

Python: dekoratori

- Python štiti prikazani član tako što interno menja njegovo ime tako da ono uključuje ime klase.
  - Sada se pristup atributu može izvesti kao: *object._className__attrName*.
- Izlaz izvršavanja prethodnog koda će biti 1, 2, 2, repsektivno ako poslednju liniju tog koda zamenimo sa:

```
print (counter._JustCounter__secretCount)
```

```
1
2
2
```

Dekoratori @classmethod i @staticmethod

Za korišćenje zajedničkih metoda klase u ranijim verzijama se koristila funkcija koja je metodu "prevodila" u statičku ili klasnu (npr. nakon definicije funkcije f unutar date sledilo je f=classmethod(f) ili f=staticmethod(f)).
Sada se koriste dekoratori.

```
class Roditelj:
  @staticmethod
  def staticki_pozdrav(): print("Roditeljski staticki pozdrav !")
  @classmethod
  def klasni_pozdrav(cls):
     if(cls.__name__=="Sin"):          print("Zdravo sine !")
     elif(cls.__name__=="Kcerka"):     print("Zdravo kceri !")
     elif(cls.__name__=="Roditelj"): print("Roditeljski klasni
pozdrav !")
     else: print("Zdravo ???")
```

Python: dekoratori

```
class Sin(Roditelj):
  def PozdravPrekoObjekta(self):
        print("Sin pozdravlja")

  @staticmethod
  def staticki_pozdrav():
        print("Sinov staticki pozdrav !")

  @classmethod
  def klasni_pozdrav(cls):
       if(cls.__name__=="Sin"):
              print("Sinov klasni pozdrav !")
       else:
              print("Neka je druga klasa");

class SinovSin(Sin):
       pass

class Kcerka(Roditelj):
  def PozdravPrekoObjekta(self):
        print("Kcerka pozdravlja")
```

Python: dekoratori

```
Roditelj.staticki_pozdrav() #Roditeljski staticki pozdrav !
Roditelj.klasni_pozdrav()   #Roditeljski klasni pozdrav !
roditelj = Roditelj()       #
roditelj.staticki_pozdrav() #Roditeljski staticki pozdrav !
roditelj.klasni_pozdrav()   #Roditeljski klasni pozdrav !
print()
Sin.staticki_pozdrav()      #Sinov staticki pozdrav !
Sin.klasni_pozdrav()        #Sinov klasni pozdrav !
sin = Sin()                 #
sin.staticki_pozdrav()      #Sinov staticki pozdrav !
sin.klasni_pozdrav()        #Sinov klasni pozdrav !
print()
SinovSin.staticki_pozdrav() #Sinov staticki pozdrav !
SinovSin.klasni_pozdrav()   #Neka je druga klasa
print()
Kcerka.staticki_pozdrav()   #Roditeljski staticki pozdrav !
Kcerka.klasni_pozdrav()     #Zdravo kceri !
kcerka = Kcerka()           #
kcerka.staticki_pozdrav()   #Roditeljski staticki pozdrav !
kcerka.klasni_pozdrav()     #Zdravo kceri !
```

Python: regularni izrazi

Regularni izrazi
predstavljaju specijalnu sekvencu karaktera (uzorak, pattern) koji pomažu da se nađe string ili skup stringova koji odgovaraju kriterijumu predstavljenom na ovaj način.

- Modul **re** (**r**egular-**e**xpression) omogućuje punu podršku regularnih izraza u Python-u.
- Funkcija re.match() pokušava da pronađe poklapanje na početku stringa sa pattern-om:

```
re.match(pattern, string, flags=0)
```

  - pattern regularni izraz za poklapanje.
  - string string koji se pretražuje u smislu poklapanja sa datim uzorkom.
  - flags modifikatori.

  poklapanje može biti uspešno ili neuspešno.

Python: re, match

- Funkcija *re.match* vraća **match** objekat ako je uspešno poklapanje ili **None** ako poklapanje nije uspešno.
- Za uzimanje rezultata poklapanja **match** objekta koriste se funkcije:
  - group(num=0), vraća rezultat poklapanja za datu grupu (ili specifičnog podgrupnog broja - num)
  - groups(), vraća sve podgrupe koje se poklapaju sa uzorkom kao n-torku (praznu ako nema poklapanja)

```
import re
line = "Cats are smarter than dogs"
matchObj = re.match( r'(.*) are (.*?) .*', line, re.M|re.I)
if matchObj:
   print ("matchObj.group( ) : ", matchObj.group( ))
   print ("matchObj.group(1) : ", matchObj.group(1))
   print ("matchObj.group(2) : ", matchObj.group(2))
else:
   print ("No match!!")
```

```
matchObj.group( ) : Cats are smarter than dogs
matchObj.group(1) : Cats
matchObj.group(2) : smarter
```

Python: re, search

- Funkcija *search* traži prvo pojavljivanje uzorka (*pattern*) unutar stringa (*string*) sa opcionalnim zastavicama (*flags*).

```
re.search(pattern, string, flags=0)
```

pattern,	regularni izraz koji predstavlja uzorak za poklapanje.
string,	string u kom se traži poklapanje sa uzorkom.
flags,	modifikatori.

- Funkcija *re.search()* ako je uspešna vraća **match** objekat ili **None** ako ne nađe uzorak.

```
import re
line = "Cats are smarter than dogs"
searchObj = re.search( r'(.*) are (.*?) .*', line, re.M|re.I)
if searchObj:
   print ("searchObj.group( ) : ", searchObj.group())
   print ("searchObj.group(1) : ", searchObj.group(1))
   print ("searchObj.group(2) : ", searchObj.group(2))
else:
   print ("Nothing found!!")
```

```
matchObj.group( ) : Cats are smarter than dogs
matchObj.group(1) : Cats
matchObj.group(2) : smarter
```

Python: re.match, re.search, re.sub

- Razlika između search i match je u tome što u uzorku **match** proverava poklapanje samo na početku stringa dok **search** proverava poklapanje bilo gde u stringu.

```
import re
line = "Cats are smarter than dogs"

matchObj = re.match( r'dogs', line, re.M|re.I)
if matchObj:
   print ("match --> matchObj.group() : ", matchObj.group() )
else:
   print ("No match!!")

searchObj = re.search( r'dogs', line, re.M|re.I)
if searchObj:
   print ("search --> searchObj.group() : ", searchObj.group() )
else:
   print ("Nothing found!!")
```

```
No match!!
search --> matchObj.group() : dogs
```

- Jedna od bitnih metoda je sub (substitution) koja zamenjuje sve pojave uzorka ili do maksimalno zadatog broja zamena:

```
re.sub(pattern, repl, string, max=0)
```

ovaj metod vraća modifikovani string.

Python: re.sub, modifikatori

```
import re
phone = "2004-959-559 # This is Phone Number"
# Delete Python-style comments
num = re.sub(r'#.*$', "", phone)
print ("Phone Num : ", num)
# Remove anything other than digits
num = re.sub(r'\D', "", phone)
print ("Phone Num : ", num)
```

```
Phone Num :  2004-959-559
Phone Num :  2004959559
```

Modifikator	Opis
re.I	case-**i**nsensitive poklapanje.
re.L	Interpretira reči prema tekućoj **l**okalizaciji (utiče na alfabetsku grupu \w \W te na \b \B).
re.M	Multilinijsko poklapanje (utiče na ^ $).
re.S	Znak tačka (dot, period) poklapa svaki karakter uključujući i *newline*.
re.U	Interpretira slova prema Unicode-u (utiče na \w, \W, \b, \B, \d \D, \s \S).
re.X	Bolja čitljivost uzorka jer ignoriše beline (osim ako su navedene u uzorku) i omogućuje stavljanje komentara (#) u uzorak.

Python: uzorci

Uzorak	Opis
^	Poklapanje na početku linije.
$	Poklapanje od kraju linije.
.	Poklapanje bilo kog karaktera osim newline (osim ako je re.S).
[...]	Poklapanje bilo kog navedenog karaktera.
[^...]	Nepoklapanje bilo kog navedenog karaktera.
n?	Poklapanje sa stringom koji sadrži 0 ili jedan n.
n*	Poklapanje sa stringom koji sadrži 0, 1 ili više pojavljivanja n.
n+	Poklapanje sa stringom koji sadrži bar jedan n.
n{ x}	Pronalazi string koji sadrži sekvencu od x n-ova.
n{ min,}	Pronalazi string koji sadrži sekvencu od bar min n-ova.
n{ min, max}	Pronalazi string koji sadrži sekvencu od min do max n-ova.
a| b	Poklapanje sa a ili b.
(re)	Poklapanje sa grupom regularnih izraza.
(?imx)	Privremeno menja (uključenjem) i, m, ili x opciju unutar regularnog izraza.

Python: re, uzorci

Uzorak	Opis
(?-imx)	Privremeno menja (isključenjem) opcije i, m ili x unutar regularnog izraza.
(?: re)	Kreira non-capturing grupu. npr. (?:abc){3} -> abcabcabc. Nema grupa.
(?#...)	Postavljanje komentara unutar regularnog izraza npr. \d(?#digit)
(?= n)	Pronalazi string iza koga sledi n.
(?! n)	Pronalazi string iza koga ne sledi n.
(?> re)	Atomska grupa, optimizuje traženje, npr. (?>his|this) u reči 'smashing' ako ne nađe ***his*** neće tražiti ni ***this.***
\w	Nađi slovo.
\W	Nađi što nije slovo.
\s	Whitespace, isto kao [\t\n\r\f].
\S	Nonwhitespace.
\d	Nađi cifru (isto kao [0-9]).

Python: re uzorci

Uzorak	Opis
\D	Nađi što nije cifra.
\A	Poklapanje samo na početku stringa.
\Z	Poklapanje samo na kraju stringa (za newline proverava pre njega)
\z	Poklapanje na kraju stringa.
\G	Nastavlja proveru gde je prethodna provera stala. pr. \G\w u 'this this' daće ponavljanjem uzorka **t h i s failed**.
\b	Nađi poklapanje na početku ili kraju reči.
\B	Nađi što nema poklapanje na početku ili kraju reči.
\n, \t,	Poklapanje sa *newlines*, *tabs*, itd.
\1...\9	Poklapanje n-te grupe podizraza.

- Primeri regularnih izraza:

```
a = re.compile(r"""\d+ # the integral part
                   \.  #the decimal point
                   \d* # some fractional digits""", re.X)
b = re.compile(r"\d+\.\d*") #VERBOSE
```

  - python poklapanje: "python".
  - [Pp]ython poklapanje: "Python" ili "python"
  - rub[ye] poklapanje: "ruby" ili "rube"

Python: re uzorci

- [aeiou] poklapanje: bilo koji mali samoglasnik
- [0-9] poklapanje: bilo koja cifra; isto kao [0123456789]
- [a-z] poklapanje: bilo koje malo ASCII slovo
- [A-Z] poklapanje: bilo koje veliko ASCII slovo
- [a-zA-Z0-9] poklapanje: bilo šta od navedenog
- [^aeiou] poklapanje: bilo šta različito od malih samoglasnika
- [^0-9] poklapanje: bilo šta različito od cifre

- Specijalni karakteri:
  - . poklapanje: bilo koji karakter osim *newline* (osim ako je dat re.S)
  - \d isto kao [0-9]
  - \D isto kao [^0-9]
  - \s poklapanje praznina, isto kao [\t\r\n\f]
  - \S poklapanje nepraznina, isto kao [^ \t\r\n\f]
  - \w poklapanje slova, isto kao [A-Za-z0-9_]
  - \W nađi što nije slovo, isto kao [^A-Za-z0-9_]

Python: re uzorci

- Ponavljanja:
  - ruby? poklapanje: "rub", "ruby"
  - ruby* poklapanje: "rub", "ruby", "rubyy", ...
  - ruby+ poklapanje: "ruby", "rubyy", ...
  - \d{3} poklapanje: tačno 3 cifre
  - \d{3,} poklapanje: 3 ili više cifara
  - \d{3,5} poklapanje: 3, 4 ili 5 cifara
- Pohlepno i nepohlepno poklapanje
  - <.*> Pohlepno (greedy) ponavljanje: poklapanje "<**python>perl**>"
  - <.*?> Nongreedy: matches "<python>" in "<**python**>perl>"
- Grupisanje sa zagradama:
  - \D\d+ bez grupe: + ponavlja \d
  - (\D\d)+ grupisano: + ponavlja par \D\d
  - ([Pp]ython(,)?)+ poklapanje: "Python", "Python, python, python", ...
- Povratne reference (*backreferences*):
  - ([Pp])ython&\1ails poklapanje: python&pails ili Python&Pails
  - (['"])[^\1]*\1 string sa apostrofima ili navodnicima.
    \1 odnosi se na prvu grupu

Python: re uzorci

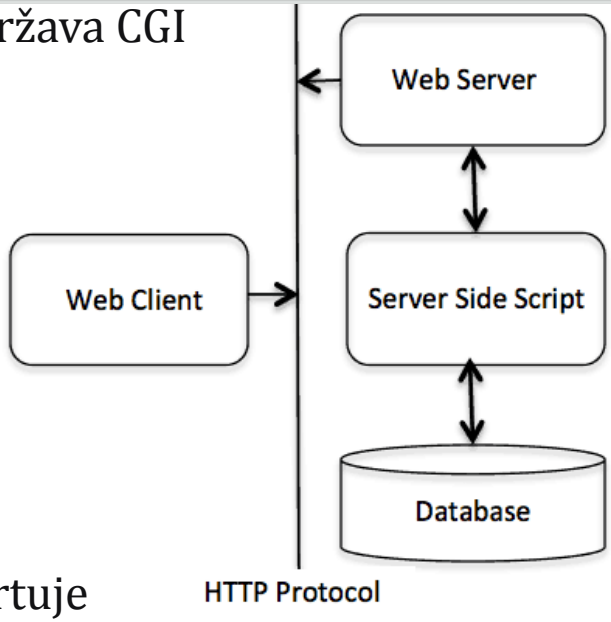
- Alternative:
  - python|perl — poklapanje: "python" ili "perl"
  - rub(y|le) — poklapanje: "ruby" ili "ruble"
  - Python(!+|\?) — poklapanje: iza "Python" sledi jedan ili više ! ili jedan ?
- Ankeri (specificiraju poziciju poklapanja):
  - ^Python — poklapanje: "Python" je na početku stringa.
  - Python$ — poklapanje: "Python" je na kraju stringa.
  - \APython — poklapanje: "Python" je **samo** na početku stringa.
  - Python\Z — poklapanje: "Python" je **samo** na kraju stringa.
  - \bPython\b — poklapanje: "Python" je unutar reči.
  - \brub\B — poklapanje: "rub" npr. u reči "rube" i "ruby"
  - Python(?=!) — poklapanje: "Python", ako je iza znak !
  - Python(?!!) — poklapanje: "Python", ano iza nema znak !
- Posebna sintaksa sa zagradama:
  - R(?#comment) — poklapanje: "R". Sve ostalo je komentar.
  - R(?i)uby — case-**i**nsensitive dok se traži "uby"
  - R(?i:uby) — kao iznad

Python: CGI

- Common Gateway Interface (CGI) je skup standarda koji definišu kako se informacije razmenjuju između web server-a i korisničkog skript-a.
- Primer: kada korisnik klikne na hiperlink da bi pregledao određenu web stranu ili URL dešava se sledeće:
  - browser kontaktira HTTP web server i zahteva URL (ime fajla).
  - Web server parsira URL i traži ime fajla.
    - ako nađe zahtevani fajl onda ga šalje nazad ka browser-u,
    - ako ne nađe zahtevani fajl onda šalje poruku o grešci indicirajući da se tražio pogrešan fajl.
  - Web browser uzima odgovor od web server-a i prikazuje ili primljeni fajl ili poruku o grešci.
  - Međutim, moguće je podesiti HTTP server tako da ako se traži fajl u nekom datom direktorijumu da se fajl ne prosleđuje nazad nego da se umesto toga taj program izvrši i da se rezultat tog programa vrati nazad browser-u i da se isti prikaže.
  - Ova funkcija se naziva Common Gateway Interface (CGI) a programi se zovu CGI skriptovi (Python Script, PERL Script, Shell Script, C, C++,...).

Python: CGI

- Za Web Server prvo se proveri da li Web Server podržava CGI i da li je konfigurisan da rukuje CGI programima.
- Svi CGI programi koje izvršava HTTP server su smešteni u prekonfigurisani direktorijum koji se naziva CGI Directory i po konvenciji to je /var/www/cgi-bin.
- Po konvenciji CGI fajlovi imaju ekstenziju .cgi, ali može se ostaviti da Python-ovi fajlovi imaju ekstenziju .py.



- Podrazumevano Linux server je konfigurisan da startuje samo skriptove u direktorijumu cgi-bin u /var/www (za specifikaciju nekog drugog direktorijuma izvrše se izmene u httpd.conf fajlu)

```
<Directory "/var/www/cgi-bin">
  AllowOverride None
  Options ExecCGI
  Order allow,deny
  Allow from all
</Directory>
<Directory "/var/www/cgi-bin">
  Options All
</Directory>
```

Python: CGI

- Sledi primer CGI programa (hello.py) koji se nalazi u /var/www/cgi-bin direktorijumu i koji će se zahtevati jednostavnim linkom iz browser-a:

```
#!/usr/bin/python
print ("Content-type:text/html\r\n\r\n")
print ('<html>' )
print ('<head>' )
print ('<title>Hello Word - CGI Program</title>' )
print ('</head>' )
print ('<body>' )
print ('<h2>Hello Word! This is CGI program</h2>' )
print ('</body>' )
print ('</html>' )
```

Hello Word! This is CGI program

- Prva linija koda kada stigne do browser-a specificira da se sadržaj ispiše na ekran (Content-type:text/html\r\n\r\n). Ova linija je deo HTTP zaglavlja koje se šalje browser-u da shvati koji se sadržaj šalje.
- HTTP zaglavlje ima sledeću formu:

HTTP **Field Name:** Field Content

npr. **Content-type:** text/html\r\n\r\n

Python: CGI

- Neka značajna HTTP zaglavlja (header-i):

Header	Opis
`Content-type:`	MIME string koji definiše format fajla koji se vraća. Npr. Content-type:text/html
`Expires: Date`	Koristi se da browser odluči kada da se osveži stranica. Validni format je npr. 01 Jan 1998 12:00:00 GMT.
`Location: URL`	URL koji se vraća umesto zahtevanog URLa. Koristi se za redirekciju zahteva za fajlom.
`Last-modified: Date`	Vreme poslednje modifikacije resursa.
`Content-length: N`	Dužina podataka u bajtovima koja će biti vraćena. Koristeći ovu informaciju browser može dati procenjeno vreme za učitavanje.
`Set-Cookie: String`	Postavljanje cookie-a prosleđenog kao string.

Python: CGI environment variables

Variabla	Opis
CONTENT_TYPE	Tip podataka sadržaja. Koristi se npr. kada klijent šalje attach-ovan sadržaj ka serveru (npr. file upload).
CONTENT_LENGTH	Dužina upita za POST zahteve.
HTTP_COOKIE	Vraća skup cookie-a u formi key-value.
HTTP_USER_AGENT	Naziv WEB browser-a.
PATH_INFO	Staza CGI skripta.
QUERY_STRING	URL-kodovana informacija koja se šalje sa GET zahtevom.
REMOTE_ADDR	IP adresa klijenta.
REMOTE_HOST	Naziv hosta klijenta.
REQUEST_METHOD	Metod zahteva (npr. GET, POST).
SCRIPT_FILENAME	Puna staza CGI skripta.
SCRIPT_NAME	Naziv CGI skripta.
SERVER_NAME	Naziv servera (hostname ili IP adresa).
SERVER_SOFTWARE	Naziv i verzija softvera na strani servera.

Python: CGI, get, post

- Program koji izlistava sve CGI varijable:

```
#!/usr/bin/python
import os
print ("Content-type: text/html\r\n\r\n")
print ("<font size=+1>Environment</font><\br>")
for param in os.environ.keys():
   print ("<b>%20s</b>: %s<\br>" % (param, os.environ[param]))
```

GET i POST metode

- Slanje informacije GET metodom:
  - GET je podrazumevani metod koji šalje kodovane korisničke informacije dodate zahtevu za stranicom u formi parova ključ=vrednost (name=value) sa delimiterom & koje se vide u adresnom baru.
  - U zahtevu su nakon navoda stranice ove informacije odvojene znakom ?

    `http://www.test.com/cgi-bin/hello.py?key1=value1&key2=value2`
  - Ovim metodom se ne šalju osetljive informacije (npr. password) ka serveru.
  - GET metod ima ograničenje u količini podataka: 2-8 KB po zahtevu.
  - GET metod šalje infromacije korišćenjem QUERY_STRING zaglavlja (dostupno preko QUERY_STRING promenljive okruženja).

Python: CGI, get, post

- Primer: slanje vrednosti metodom GET:

  /cgi-bin/hello_get.py?first_name=ZARA&last_name=ALI

program hello_get.py:

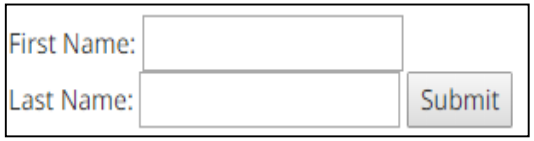
```
#!/usr/bin/python
# Import modules for CGI handling
import cgi, cgitb
#Traceback      Create instance of FieldStorage
cgitb.enable(); form = cgi.FieldStorage()
# Get data from fields
first_name = form.getvalue('first_name')
last_name = form.getvalue('last_name')
print ("Content-type:text/html\r\n\r\n" )
print ("<html>" )
print ("<head>" )
print ("<title>Hello - Second CGI Program</title>" )
print ("</head>" )
print ("<body>" )
print ("<h2>Hello %s %s</h2>" % (first_name, last_name) )
print ("</body>" )
print ("</html>")
```

Hello ZARA ALI

Python: CGI, forme

- Slanje vrednosti može se obaviti i HTML formom:

```
<form action="/cgi-bin/hello_get.py" method="get">
  First Name: <input type="text" name="first_name"> <br />
  Last Name: <input type="text" name="last_name" />
  <input type="submit" value="Submit" />
</form>
```



- POST metod šalje informacije kao odvojenu poruku. Program hello_get.py je isti kao na prethodnom slajdu, samo je sada HTML forma sa navedenom POST metodom. Izgled forme se ovim ne menja.

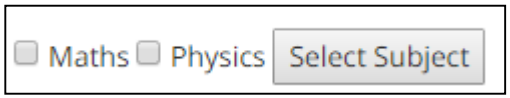
```
<form action="/cgi-bin/hello_get.py" method="post">
  First Name: <input type="text" name="first_name"><br />
  Last Name: <input type="text" name="last_name" />
  <input type="submit" value="Submit" />
</form>
```

- Slanje checkbox podataka je kao što sledi (npr. action="http://127.0.0.1/cgi-bin/mptest_7p22str.py"):

```
<form action="/cgi-bin/checkbox.cgi" method="POST" target="_blank">
  <input type="checkbox" name="maths" value="on" /> Maths
  <input type="checkbox" name="physics" value="on" /> Physics
  <input type="submit" value="Select Subject" />
</form>
```

Python: CGI, forme

- Sledi kod checkbox.cgi skripta koji rukuje ulaznim podacima koji se dobijaju od web browser-a za checkbox dugmad.

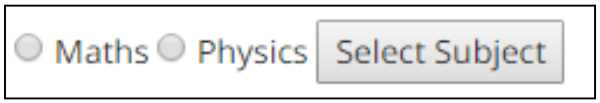



```
#!/usr/bin/python
# Import modules for CGI handling
import cgi, cgitb
#Traceback      Create instance of FieldStorage
cgitb.enable(); form = cgi.FieldStorage()
# Get data from fields
if form.getvalue('maths'):   math_flag = "ON"
else:                        math_flag = "OFF"
if form.getvalue('physics'): physics_flag = "ON"
else:                        physics_flag = "OFF"
print ("Content-type:text/html\r\n\r\n")
print ("<html>" )
print ("<head>" )
print ("<title>Checkbox - Third CGI Program</title>" )
print ("</head>" )
print ("<body>" )
print ("<h2> CheckBox Maths is : %s</h2>" % math_flag )
print ("<h2> CheckBox Physics is : %s</h2>" % physics_flag )
print ("</body>" )
print ("</html>")
```

Python: CGI, forme

- Slanje infomacija o radio dugmadima (radio button):

```
<form action="/cgi-bin/radiobutton.py" method="post" target="_blank">
  <input type="radio" name="subject" value="maths" /> Maths
  <input type="radio" name="subject" value="physics" /> Physics
  <input type="submit" value="Select Subject" />
</form>
```



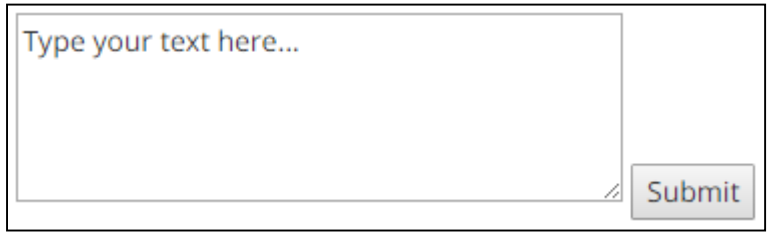
- Pripadni kod radiobutton.py skripta je kao što sledi:

```
#!/usr/bin/python
# Import modules for CGI handling
import cgi, cgitb
#Traceback      Create instance of FieldStorage
cgitb.enable(); form = cgi.FieldStorage()
# Get data from fields
if form.getvalue('subject'): subject = form.getvalue('subject')
else:                        subject = "Not set"
print ("Content-type:text/html\r\n\r\n" )
print ("<html>"); print ("<head>" )
print ("<title>Radio - Fourth CGI Program</title>" )
print ("</head>" ); print ("<body>")
print ("<h2> Selected Subject is %s</h2>" % subject )
print ("</body>"); print ("</html>" )
```

Python: CGI, forme

- Forma za slanje TEXTAREA (višelinijskog) sadržaja je kao što sledi:

```
<form action="/cgi-bin/textarea.py" method="post" target="_blank">
  <textarea name="textcontent" cols="40" rows="4">
    Type your text here...
  </textarea>
<input type="submit" value="Submit" />
</form>
```



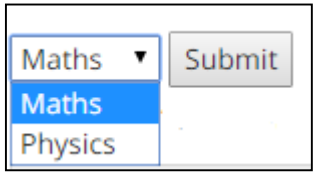
- Pripadni kod za rukovanje podacima textarea je kao što sledi:

```
#!/usr/bin/python
# Import modules for CGI handling
import cgi, cgitb
#Traceback      Create instance of FieldStorage
cgitb.enable(); form = cgi.FieldStorage()
# Get data from fields
if form.getvalue('textcontent'):
   text_content = form.getvalue('textcontent')
else: text_content = "Not entered"
print ("Content-type:text/html\r\n\r\n");print("<html>");print("<head>")
print ("<title>Text Area - Fifth CGI Program</title>"); print("</head>")
print ("<body>")
print ("<h2> Entered Text Content is %s</h2>" % text_content)
print ("</body>"); print ("</html>")
```

Python: CGI, forme

- Slanje podatka iz selekcione liste (drop-down box):

```
<form action="/cgi-bin/dropdown.py" method="post" target="_blank">
  <select name="dropdown">
    <option value="Maths" selected>Maths</option>
    <option value="Physics">Physics</option>
  </select>
  <input type="submit" value="Submit"/>
</form>
```



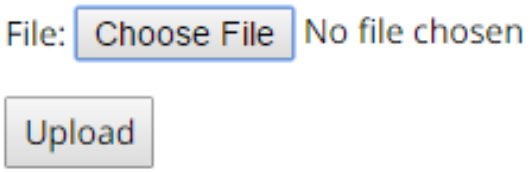
- Pripadni kod za prihvatanja i obradu podatka odabranog iz selekcione liste je:

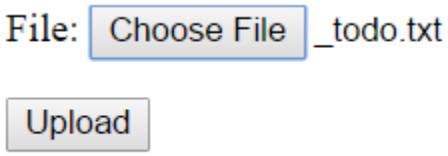
```
#!/usr/bin/python
import cgi, cgitb
cgitb.enable(); form = cgi.FieldStorage()
if form.getvalue('dropdown'): subject = form.getvalue('dropdown')
else: subject = "Not entered"
print ("Content-type:text/html\r\n\r\n")
print ("<html>")
print ("<head>")
print ("<title>Dropdown Box - Sixth CGI Program</title>")
print ("</head>")
print ("<body>")
print ("<h2> Selected Subject is %s</h2>" % subject)
print ("</body>")
print ("</html>")
```

Python: CGI, upload

- Uploadovanje fajlova se obavlja tako da HTML forma ima postavljen atribut **enctype=multipart/form-data**.
- Input tag tipa file type kreira dugme "Browse".

```
<html>
<body>
<form enctype="multipart/form-data"
      action="http://127.0.0.1/cgi-bin/save_file.py" method="post">
  <p>File: <input type="file" name="filename" ></p>
  <p><input type="submit" value="Upload" ></p>
</form>
</body>
</html>
```





Python: CGI, upload

- Sledi skript **save_file.py** koji rukuje uploadom fajla:

```
#!C:/Users/Pero/AppData/Local/Programs/Python/Python35/python.exe
import cgi, os
import cgitb # Traceback manager.
cgitb.enable()
form = cgi.FieldStorage()
# Get filename here.
fileitem = form['filename']
# Test if the file was uploaded
if fileitem.filename:
  # strip leading path from file name to avoid
  # directory traversal attacks
  fn = os.path.basename(fileitem.filename)
  fp=open( fn, 'wb'); fp.write(fileitem.file.read()); fp.close()
  message = 'The file ' + fn + ' was uploaded successfully'
else:
  message = 'No file was uploaded'
print ("Content-Type: text/html\r\n\r\n <html> <body> <p>%s</p> </body>
</html> " % (message,))
```

- Na Linux-u bi trebalo postaviti gore navedenu liniju koda za funkciju open kao:
  `fn = os.path.basename(fileitem.filename.replace("\\", "/" ))`

Download fajla

```
#!C:/Users/Peeet/AppData/Local/Programs/Python/Python35/python.exe

import cgitb
cgitb.enable()

print ("Content-Type: text/plain")
print ("Content-Disposition:attachment;filename=_dovucen_fajl.txt")
print ()

filename = "_todo.txt"
f = open(filename, 'r')
for line in f:
    print (line)
f.close()
```