

Programski jezik Python

Šantić, Tomislav

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:200:374473>

Rights / Prava: [In copyright / Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2024-05-20**

Repository / Repozitorij:

[Faculty of Electrical Engineering, Computer Science
and Information Technology Osijek](#)



SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE RAČUNARSTVA I INFORMACIJSKIH
TEHNOLOGIJA

Stručni studiji informatike

PROGRAMSKI JEZIK PYTHON

Završni rad

Tomislav Šantić

Osijek, 2017.



FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Osijek, 27.02.2017.

Odboru za završne i diplomske ispite

**Imenovanje Povjerenstva za obranu završnog rada
na preddiplomskom stručnom studiju**

Ime i prezime studenta:	Tomislav Šantić
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	A4159, 22.09.2011.
OIB studenta:	99512557867
Mentor:	Doc.dr.sc. Damir Blažević
Sumentor:	
Predsjednik Povjerenstva:	Doc.dr.sc. Ivan Aleksić
Član Povjerenstva:	Dr.sc. Ivan Vidović
Naslov završnog rada:	Programski jezik Python
Znanstvena grana rada:	Programsko inženjerstvo (zn. polje računarstvo)
Zadatak završnog rada	Opisati povjesni razvoj, značajke i primjenu programskog jezika Python. Opisati sintaksu, dostupna razvojna okruženja i mogućnosti. Izraditi nekoliko aplikacija i prikazati njihov razvoj od algoritma, programskog koda, izvršne datoteke do paketa namijenjenoga za distribuciju. Sve prikazati na Linux/Ubuntu platformi koristeći aplikacije otvorenog koda.
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3Postignuti rezultati u odnosu na složenost zadatka: 2Jasnoća pismenog izražavanja: 3Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	27.02.2017.
Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis: Datum:



FERIT

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 19.03.2017.

Ime i prezime studenta:	Tomislav Šantić
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	A4159, 22.09.2011.
Ephorus podudaranje [%]:	7

Ovom izjavom izjavljujem da je rad pod nazivom: **Programski jezik Python**

izrađen pod vodstvom mentora Doc.dr.sc. Damir Blažević

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD.....	1
1.1 Zadatak završnog rada.....	1
1.2 Povijesni razvoj programskog jezika Python.....	1
1.3 Instalacija Python unutar Linux okruženja.....	1
2. PYTHON.....	3
2.1. Varijable i tipovi varijabli.....	4
2.1.1. Pridruživanje vrijednosti varijablama.....	4
2.1.2 Brojevi.....	5
2.1.3. Stringovi.....	5
2.1.4. Liste.....	6
2.1.5. Rječnici.....	6
2.2 Operatori.....	6
2.2.1 Aritmetički operatori.....	7
2.2.2. Operatori uspoređivanja.....	7
2.2.3. Operatori pridruživanja.....	8
2.2.4. Logički operatori.....	9
2.2.5. Operatori identiteta.....	9
2.2.6. Prioriteti operacija.....	9
2.3. Odlučivanje.....	10
2.4. Petlje.....	11
2.4.1. For petlja.....	12
2.4.2. While petlja.....	13
2.5. Funkcije.....	15
2.5.1. Definiranje funkcija.....	15
2.5.2. Funkcijski poziv.....	16
2.6. Klase i objekti.....	17
3. PRIMJERI APLIKACIJA U PROGRAMSKOM JEZIKU PYTHON.....	18
3.1. Kivy aplikacija za crtanje.....	18
3.2. Kivy kalkulator.....	21
3.3 Distribucija i stvaranje .deb datoteke.....	25
4. ZAKLJUČAK.....	27
5. LITERATURA.....	28
SAŽETAK.....	29
ŽIVOTOPIS.....	31
PRILOG A. Izvorni kod aplikacije za crtanje.....	32

1. UVOD

1.1 Zadatak završnog rada

Opisati povjesni razvoj, značajke i primjenu programskog jezika Python. Opisati sintaksu, dostupna razvojna okruženja i mogućnosti. Izraditi nekoliko aplikacija i prikazati njihov razvoj od algoritma, programskog koda, izvršne datoteke do paketa namijenjenoga za distribuciju također prikazati na Linux/Ubuntu platformi.

1.2 Povijesni razvoj programskog jezika Python

Python nastaje u kasnim osamdesetim godinama od strane nizozemskog programera Guido van Rossuma koji je imao viziju stvaranja novog jezika koji će biti jednostavan i ugodan za korištenje. Python za cilj ima kvalitetu, produktivnost, portabilnost i modularnost. Kvaliteta se postiže lako čitljivošću i samom duljinom programskog koda, što implicira produktivnost jer programer mora pisati, ispravljati i održavati manje koda. Portabilnost je omogućena zbog toga što se Python program može pokretati neovisno o operacijskom sutavu, a modularnost se postiže korištenjem raznih biblioteka koje se mogu iskoristiti za proširenje funkcionalnosti.

1.3 Instalacija Python unutar Linux okruženja

Obzirom da je Python programski jezik moguće podesiti na različitim operacijskim sustavima, instalacija je prikazana unutar Debian 8.6.0 distribucije Linuxa. Iako Python 2.7.x dolazi automatski podešen prilikom instalacije Debian operacijskog sustava, za Python 3.x.x. verziju potrebna je instalacija. U tablici 1.1. prikazani su koraci instalacije programskog jezika Python.

Tablica 1.1. Koraci instalacije programskog jezika Python

1. korak	Otvoriti terminal
2. korak	Upisati sljedeću naredbu bez navodnika i pritisnuti tipku ENTER: “su”
3. korak	Unesite zaporku “root” korisnika
4. korak	Upisati naredbu: “apt-get install build-essential libncursesw5-dev libreadline5-dev libssl-dev”
5. korak	Upisati sljedeću naredbu bez navodnika i pritisnuti tipku ENTER: “apt-get install libgdbm-dev libc6-dev libsqlite3-dev tk-dev”
6. korak	Upisati sljedeću naredbu bez navodnika i pritisnuti tipku ENTER: “wget http://www.python.org/ftp/python/python/3.x/Python-3.x.tar.bz2”

7. korak	Upisati sljedeću naredbu bez navodnika i pritisnuti tipku ENTER: “tar -xjf Python-3.xtar.bz2 cd Python-3.x”
8. korak	Upisati sljedeću naredbu bez navodnika i pritisnuti tipku ENTER: “./configure -prefix=/opt/python3”
9. korak	Upisati sljedeću naredbu bez navodnika i pritisnuti tipku ENTER: “make”
10. korak	Upisati sljedeću naredbu bez navodnika i pritisnuti tipku ENTER: “make install”

2. PYTHON

Obzirom da postoje brojni programski jezici, opravdano je pitanje: "Zašto Python?". Python je jezik koji za cilj ima kvalitetu, produktivnost, portabilnost, modularnost i ugodnost korištenja. Kvaliteta se postiže lakom čitljivošću programskog koda, produktivnost se unaprijeđuje smanjenjem količine napisanog koda potrebnog za postizanje rezultata, portabilnost je omogućena zbog toga što se Python program može pokretati neovisno o operacijskom sustavu, a modularnost se postiže korištenjem raznih biblioteka koje se mogu iskoristiti za proširenje funkcionalnosti. Obzirom da je ugodnost korištenja subjektivna mjera, nužan je primjer kako bi se to moglo demonstrirati.



```
1 def fibonacci(unos):
2     pocetak_sekvence, sljedeci_broj_sekvence = 0 , 1
3     while pocetak_sekvence < unos:
4         print "%d" % pocetak_sekvence
5         pocetak_sekvence, sljedeci_broj_sekvence = sljedeci_broj_sekvence, ( pocetak_sekvence + sljedeci_broj_sekvence )
6
7 fibonacci(1000)
8
9 File Edit View Search Terminal Help
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
819
820
821
822
823
824
825
826
827
827
828
829
829
829
830
831
832
833
834
835
836
837
837
838
839
839
839
840
841
842
843
844
845
846
846
847
848
848
848
849
850
851
852
853
854
855
856
857
857
858
859
859
859
860
861
862
863
864
865
866
866
867
868
868
868
869
870
871
872
873
874
875
876
876
877
878
878
878
879
880
881
882
883
884
885
886
886
887
888
888
888
889
889
889
890
891
892
893
894
895
895
896
896
896
897
898
898
898
899
899
899
900
901
902
903
904
905
905
906
907
907
907
908
909
909
909
910
911
912
913
913
913
914
915
915
915
916
917
917
917
918
919
919
919
920
921
922
923
923
923
924
925
925
925
926
927
927
927
928
929
929
929
930
931
931
931
932
933
933
933
934
935
935
935
936
937
937
937
938
939
939
939
940
941
941
941
942
943
943
943
944
945
945
945
946
947
947
947
948
949
949
949
950
951
951
951
952
953
953
953
954
955
955
955
956
957
957
957
958
959
959
959
960
961
961
961
962
963
963
963
964
965
965
965
966
967
967
967
968
969
969
969
970
971
971
971
972
973
973
973
974
975
975
975
976
977
977
977
978
979
979
979
980
981
981
981
982
983
983
983
984
985
985
985
986
987
987
987
988
989
989
989
990
991
991
991
992
993
993
993
994
995
995
995
996
997
997
997
998
999
999
999
1000
1001
1001
1001
1002
1003
1003
1003
1004
1005
1005
1005
1006
1007
1007
1007
1008
1009
1009
1009
1010
1011
1011
1011
1012
1013
1013
1013
1014
1015
1015
1015
1016
1017
1017
1017
1018
1019
1019
1019
1020
1021
1021
1021
1022
1023
1023
1023
1024
1025
1025
1025
1026
1027
1027
1027
1028
1029
1029
1029
1030
1031
1031
1031
1032
1033
1033
1033
1034
1035
1035
1035
1036
1037
1037
1037
1038
1039
1039
1039
1040
1041
1041
1041
1042
1043
1043
1043
1044
1045
1045
1045
1046
1047
1047
1047
1048
1049
1049
1049
1050
1051
1051
1051
1052
1053
1053
1053
1054
1055
1055
1055
1056
1057
1057
1057
1058
1059
1059
1059
1060
1061
1061
1061
1062
1063
1063
1063
1064
1065
1065
1065
1066
1067
1067
1067
1068
1069
1069
1069
1070
1071
1071
1071
1072
1073
1073
1073
1074
1075
1075
1075
1076
1077
1077
1077
1078
1079
1079
1079
1080
1081
1081
1081
1082
1083
1083
1083
1084
1085
1085
1085
1086
1087
1087
1087
1088
1089
1089
1089
1090
1091
1091
1091
1092
1093
1093
1093
1094
1095
1095
1095
1096
1097
1097
1097
1098
1099
1099
1099
1100
1101
1101
1101
1102
1103
1103
1103
1104
1105
1105
1105
1106
1107
1107
1107
1108
1109
1109
1109
1110
1111
1111
1111
1112
1113
1113
1113
1114
1115
1115
1115
1116
1117
1117
1117
1118
1119
1119
1119
1120
1121
1121
1121
1122
1123
1123
1123
1124
1125
1125
1125
1126
1127
1127
1127
1128
1129
1129
1129
1130
1131
1131
1131
1132
1133
1133
1133
1134
1135
1135
1135
1136
1137
1137
1137
1138
1139
1139
1139
1140
1141
1141
1141
1142
1143
1143
1143
1144
1145
1145
1145
1146
1147
1147
1147
1148
1149
1149
1149
1150
1151
1151
1151
1152
1153
1153
1153
1154
1155
1155
1155
1156
1157
1157
1157
1158
1159
1159
1159
1160
1161
1161
1161
1162
1163
1163
1163
1164
1165
1165
1165
1166
1167
1167
1167
1168
1169
1169
1169
1170
1171
1171
1171
1172
1173
1173
1173
1174
1175
1175
1175
1176
1177
1177
1177
1178
1179
1179
1179
1180
1181
1181
1181
1182
1183
1183
1183
1184
1185
1185
1185
1186
1187
1187
1187
1188
1189
1189
1189
1190
1191
1191
1191
1192
1193
1193
1193
1194
1195
1195
1195
1196
1197
1197
1197
1198
1199
1199
1199
1200
1201
1201
1201
1202
1203
1203
1203
1204
1205
1205
1205
1206
1207
1207
1207
1208
1209
1209
1209
1210
1211
1211
1211
1212
1213
1213
1213
1214
1215
1215
1215
1216
1217
1217
1217
1218
1219
1219
1219
1220
1221
1221
1221
1222
1223
1223
1223
1224
1225
1225
1225
1226
1227
1227
1227
1228
1229
1229
1229
1230
1231
1231
1231
1232
1233
1233
1233
1234
1235
1235
1235
1236
1237
1237
1237
1238
1239
1239
1239
1240
1241
1241
1241
1242
1243
1243
1243
1244
1245
1245
1245
1246
1247
1247
1247
1248
1249
1249
1249
1250
1251
1251
1251
1252
1253
1253
1253
1254
1255
1255
1255
1256
1257
1257
1257
1258
1259
1259
1259
1260
1261
1261
1261
1262
1263
1263
1263
1264
1265
1265
1265
1266
1267
1267
1267
1268
1269
1269
1269
1270
1271
1271
1271
1272
1273
1273
1273
1274
1275
1275
1275
1276
1277
1277
1277
1278
1279
1279
1279
1280
1281
1281
1281
1282
1283
1283
1283
1284
1285
1285
1285
1286
1287
1287
1287
1288
1289
1289
1289
1290
1291
1291
1291
1292
1293
1293
1293
1294
1295
1295
1295
1296
1297
1297
1297
1298
1299
1299
1299
1300
1301
1301
1301
1302
1303
1303
1303
1304
1305
1305
1305
1306
1307
1307
1307
1308
1309
1309
1309
1310
1311
1311
1311
1312
1313
1313
1313
1314
1315
1315
1315
1316
1317
1317
1317
1318
1319
1319
1319
1320
1321
1321
1321
1322
1323
1323
1323
1324
1325
1325
1325
1326
1327
1327
1327
1328
1329
1329
1329
1330
1331
1331
1331
1332
1333
1333
1333
1334
1335
1335
1335
1336
1337
1337
1337
1338
1339
1339
1339
1340
1341
1341
1341
1342
1343
1343
1343
1344
1345
1345
1345
1346
1347
1347
1347
1348
1349
1349
1349
1350
1351
1351
1351
1352
1353
1353
1353
1354
1355
1355
1355
1356
1357
1357
1357
1358
1359
1359
1359
1360
1361
1361
1361
1362
1363
1363
1363
1364
1365
1365
1365
1366
1367
1367
1367
1368
1369
1369
1369
1370
1371
1371
1371
1372
1373
1373
1373
1374
1375
1375
1375
1376
1377
1377
1377
1378
1379
1379
1379
1380
1381
1381
1381
1382
1383
1383
1383
1384
1385
1385
1385
1386
1387
1387
1387
1388
1389
1389
1389
1390
1391
1391
1391
1392
1393
1393
1393
1394
1395
1395
1395
1396
1397
1397
1397
1398
1399
1399
1399
1400
1401
1401
1401
1402
1403
1403
1403
1404
1405
1405
1405
1406
1407
1407
1407
1408
1409
1409
1409
1410
1411
1411
1411
1412
1413
1413
1413
1414
1415
1415
1415
1416
1417
1417
1417
1418
1419
1419
1419
1420
1421
1421
1421
1422
1423
1423
1423
1424
1425
1425
1425
1426
1427
1427
1427
1428
1429
1429
1429
1430
1431
1431
1431
1432
1433
1433
1433
1434
1435
1435
1435
1436
1437
1437
1437
1438
1439
1439
1439
1440
1441
1441
1441
1442
1443
1443
1443
1444
1445
1445
1445
1446
1447
1447
1447
1448
1449
1449
1449
1450
1451
1451
1451
1452
1453
1453
1453
1454
1455
1455
1455
1456
1457
1457
1457
1458
1459
1459
1459
1460
1461
1461
1461
1462
1463
1463
1463
1464
1465
1465
1465
1466
1467
1467
1467
1468
1469
1469
1469
1470
1471
1471
1471
1472
1473
1473
1473
1474
1475
1475
1475
1476
1477
1477
1477
1478
1479
1479
1479
1480
1481
1481
1481
1482
1483
1483
1483
1484
1485
1485
1485
1486
1487
1487
1487
1488
1489
1489
1489
1490
1491
1491
1491
1492
1493
1493
1493
1494
1495
1495
1495
1496
1497
1497
1497
1498
1499
1499
1499
1500
1501
1501
1501
1502
1503
1503
1503
1504
1505
1505
1505
1506
1507
1507
1507
1508
1509
1509
1509
1510
1511
1511
1511
1512
1513
1513
1513
1514
1515
1515
1515
1516
1517
1517
1517
1518
1519
1519
1519
1520
1521
1521
1521
1522
1523
1523
1523
1524
1525
1525
1525
1526
1527
1527
1527
1528
1529
1529
1529
1530
1531
1531
1531
1532
1533
1533
1533
1534
1535
1535
1535
1536
1537
1537
1537
1538
1539
1539
1539
1540
1541
1541
1541
1542
1543
1543
1543
1544
1545
1545
1545
1546
1547
1547
1547
1548
1549
1549
1549
1550
1551
1551
1551
1552
1553
1553
1553
1554
1555
1555
1555
1556
1557
1557
1557
1558
1559
1559
1559
1560
1561
1561
1561
1562
1563
1563
1563
1564
1565
1565
1565
1566
1567
1567
1567
1568
1569
1569
1569
1570
1571
1571
1571
1572
1573
1573
1573
1574
1575
1575
1575
1576
1577
1577
1577
1578
1579
1579
1579
1580
1581
1581
1581
1582
1583
1583
1583
1584
1585
1585
1585
1586
1587
1587
1587
1588
1589
1589
1589
1590
1591
1591
1591
1592
1593
1593
1593
1594
1595
1595
1595
1596
1597
1597
1597
1598
1599
1599
1599
1600
1601
1601
1601
1602
1603
1603
1603
1604
1605
1605
1605
1606
1607
1607
1607
1608
1609
1609
1609
1610
1611
1611
1611
1612
1613
1613
1613
1614
1615
16
```

2.1. Varijable i tipovi varijabli

Varijable su memorijske lokacije koje služe za pohranu vrijednosti, što znači da se prilikom stvaranja nove varijable u memoriji rezervira određeni dio prostora za pohranu. Ovisno o tipu podatka, alocira se veći ili manji dio memorije i na taj način je moguće pohranjivati različite tipove podataka od kojih izdvajamo brojeve, znakove i polja znakova, no dakako postoje i kompleksniji tipovi podataka s kojima Python može raditi. Važno je naglasit da je sve u Pythonu objekt, odnosno tipovi podataka predstavljeni su kao objekti određenog tipa podatka.

2.1.1. Pridruživanje vrijednosti varijablama

Tipove varijabli u Pythonu nije potrebno eksplisitno pisati jer se deklaracija događa automatski prilikom upisivanja vrijednosti u varijablu. Znak jednakosti (=) koristi se za pridruživanje vrijednosti varijablama. S lijeve strane znaka jednakosti nalazi se varijabla dok s desne, njezina vrijednost. Primjer programskog koda na slici 2.2. prikazuje pridruživanje vrijednosti varijablama.

```
1 prirodni_broj = 100
2
3 udaljenost_u_miljama = 1372.37
4
5 ime_korisnika = "Perun"
6
7 # primjer višestrukog pridruživanja
8 ime_studenta, kolegiji, godina = "Pero", "Programiranje", "2"
9
```

Sl. 2.2. Primjer pridruživanja vrijednosti

2.1.2 Brojevi

Brojevi su tipovi podataka koji se pohranjuju kao objekti numeričkog tipa. U tablici 2.1. prikazani su tipovi brojeva sa kojima je moguće raditi u programskom jeziku Python.

Tablica 2.1. *Tipovi brojeva*

INT	LONG	FLOAT	COMPLEX
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAE	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0j
-0x260	-052318172735L	-32.54e100	3e+26j
0x69	-4721885298529	70.2-E12	4.53e-7j

2.1.3. Stringovi

Kontinuirani nizovi znakova unutar navodnih znakova u Pythonu nazivaju se stringovima. Python omogućuje korištenje jednostrukih i dvostrukih navodnih znakova za označavanje strignova. Također su moguće operacije nad nizovima znakova pomoću operatora za zbrajanje i množenje.

```
1 recenica = "Python je zabavan!"  
2  
3 print recenica  
4 print recenica[0]  
5 print recenica[2:5]  
6 print recenica[2:5]  
7 print recenica * 2  
8 print recenica + " Ja cu se dodati prvoj recenici"
```

```
valsymoth@debian: ~/Python/lp$ python string.py  
Python je zabavan!  
P  
tho  
tho  
Python je zabavan!Python je zabavan!  
Python je zabavan! Ja cu se dodati prvoj recenici  
valsymoth@debian: ~/Python/lp$
```

Sl. 2.4. Operacije nad nizovima znakova

2.1.4. Liste

Liste predstavljaju složeni tip podatka. Liste sadrže podatke odvojene zarezom unutar uglatih zagrada ([]). Vrijednostima se može pristupati pomoću operatora odsjecanja, te je moguće i upravljanje pomoću operatora zbrajanja i množenja.

```
1 lista_namirnica = ["Kruh", "Mlijeko", "Jogurt"]
2 dodatne_namirnice = ["Ajvar", "Margarin", "Kit-kat"]
3
4 print(lista_namirnica)
5 print(lista_namirnica[0])
6 print(lista_namirnica[1:3])
7 print(lista_namirnica[1:])
8 print(lista_namirnica + dodatne_namirnice)
```

```
valsymoth@debian:~/Python$ python liste.py
['Kruh', 'Mlijeko', 'Jogurt']
['Mlijeko', 'Jogurt']
['Mlijeko', 'Jogurt']
['Kruh', 'Mlijeko', 'Jogurt', 'Ajvar', 'Margarin', 'Kit-kat']
valsymoth@debian:~/Python$
```

Sl. 2.5. Operacije nad listama

2.1.5. Rječnici

Rječnici u Python programskom jeziku funkcioniraju poput asocijativnih nizova u kojima postoje parovi ključa i vrijednosti. Rječnici se označavaju vitičastim zagradama ({}).

```
1 podaci_o_studentu = {}
2
3 podaci_o_studentu['ime'] = "Kristina"
4 podaci_o_studentu['prosjek'] = 4.57
5 podaci_o_studentu['godina'] = 3
6
7 print(podaci_o_studentu['ime'])
8 print(podaci_o_studentu.keys())
9 print(podaci_o_studentu.values())
10
```

```
valsymoth@debian:~/Python$ python rjecnici.py
Kristina
['ime', 'prosjek', 'godina']
['Kristina', 4.57, 3]
valsymoth@debian:~/Python$
```

Sl. 2.6. Operacije nad rječnicima

2.2 Operatori

Operatori su jezični konstrukti koji omogućavaju modifikaciju vrijednosti operanda. Postoje aritmetički, relacijski, logički, i bit operatori kao i operatori pridruživanja.

2.2.1 Aritmetički operatori

U sljedećoj tablici prikazati ćemo aritmetičke operatore pretpostavljajući da varijabla A iznosi 10, a varijabla B iznosi 20.

Tablica 2.2. *Aritmetički operatori*

OPERATOR	OPIS	PRIMJER
+	Zbrajanje	$A + B = 30$
-	Oduzimanje	$A - B = -10$
*	Množenje	$A * B = 200$
/	Dijeljenje	$B / A = 2$
%	Modulo	$A \% B = 0$
**	Eksponent	$A^{**}B = 10$
//	Cjelobrojno djeljenje	$9 // 2 = 4$

2.2.2. Operatori uspoređivanja

U sljedećoj tablici prikazati ćemo operatore uspoređivanja pretpostavljajući da su A i B bilo koja dva realna broja.

Tablica 2.3. *Operatori uspoređivanja*

OPERATOR	OPIS	PRIMJER
==	Ako su vrijednosti jednake, uvjet postaje istinit	(a == b)
!=	Ako su vrijednosti nisu jednake, uvjet postaje istinit	A != B
<>	Ako vrijednosti nisu jednake izraz postaje istinit. Sličan !=.	(a <> b)
>	Ako je vrijednost lijevog operanda veća od desnog, izraz je istinit.	A > B
<	Ako je vrijednost lijevog operanda manja od desnog, izraz je istinit.	A < B
>=	Ako je vrijednost lijevog operanda veća ili jednaka od desnog, izraz je istinit.	A >= B
<=	Ako je vrijednost lijevog operanda manja ili jednaka od desnog, izraz je istinit.	A <= B

2.2.3. Operatori pridruživanja

U sljedećoj tablici prikazati ćemo operatore pridruživanja s primjerima.

Tablica 2.4. *Operatori pridruživanja*

OPERATOR	OPIS	PRIMJER
=	Pridruživanje vrijednosti s desne strane operatora operandu s lijeva.	$A = 3 + 7$ Vrijednost A: 10
+=	Zbrajanje vrijednosti s obje strane operatora i pridruživanje lijevom operandu	$A = 3$ $A += 7$ Vrijednost A: 10
-=	Oduzimanje vrijednosti s lijeve strane operatora desnom i pridruživanje lijevom operandu	$A = 7$ $A -= 3$ Vrijednost A: 4
*=	Množenje vrijednosti s lijeve strane operatora desnom i pridruživanje lijevom operandu	$A = 10$ $A *= 5$ Vrijednost A: 50
/=	Dijeljenje vrijednosti s lijeve strane operatora desnom i pridruživanje lijevom operandu	$A = 10$ $A /= 5$ Vrijednost A: 2
%=	Modulus vrijednosti s lijeve strane operatora desnom i pridruživanje lijevom operandu	$A = 10$ $A \%= 5$ Vrijednost A: 0
**=	Potenciranje vrijednosti s lijeve strane operatora desnom, te pridruživanje lijevom operandu	$A = 10$ $A **= 2$ Vrijednost A: 100
//=	Cjelobrojno dijeljenje vrijednosti s lijeve strane operatora desnom i pridruživanje lijevom operandu	$A = 10$ $A //= 3$ Vrijednost A: 3

2.2.4. Logički operatori

U sljedećoj tablici prikazati ćemo logičke operatore pretpostavljajući da su A i B bilo koja dva objekta unutar programskog jezika Python.

Tablica 2.5. *Logički operatori*

OPERATOR	OPIS	PRIMJER
AND	Logičko I	A and B
OR	Logičko ILI	A or B
NOT	Logičko NE	not A

2.2.5. Operatori identiteta

U sljedećoj tablici prikazati ćemo operatore istinitosti pretpostavljajući da su X i Y bilo koja dva objekta unutar programskog jezika Python.

Tablica 2.6. *Operatori identiteta*

OPERATOR	OPIS	PRIMJER
is	Vraća true ako su operatori jednaki objekti unutar memorije	X is Y
is not	Vraća false ako su operatori jednaki objekti unutar memorije	X is not Y

2.2.6. Prioriteti operacija

U sljedećoj tablici prikazani su prioriteti operacija počevši s najvažnijom, odnosno onom koja se prva izvodi, do posljednje.

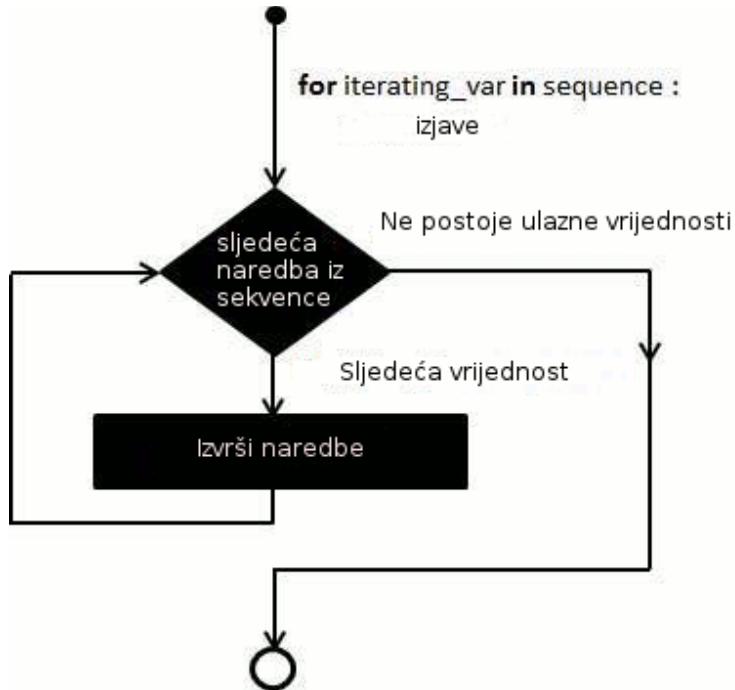
Tablica 2.7. *Prioriteti operacija*

OPERATOR	OPIS
**	EkspONENT
~ + -	KOMPLEMENT, UNARNI PLUS I UNARNI MINUS
* / % //	MNOŽENJE, DJELJENJE, MODULO I CJELOBROJNO DJELJENJE
+ -	PLUS, MINUS
>> <<	POMICANJE BITOVA U DESNO ODNOŠNO LIJEVO
&	LOGIČKO I
^	EXILI ili ILI
<= < > >=	OPERATORI USPOREĐIVANJA

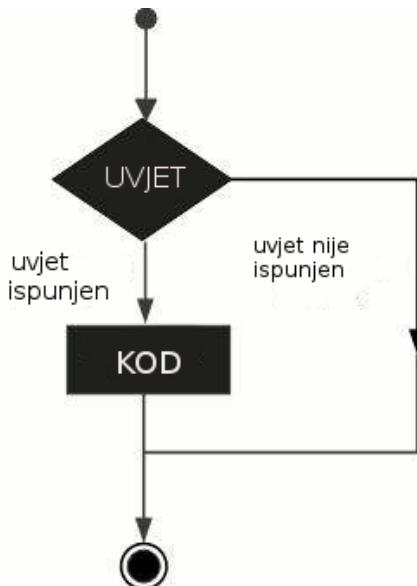
2.3. Odlučivanje

Odlučivanje unutar programskog jezika odnosi se na dio programskog koda u kojem se provjeravaju zadani izrazi te se na temelju toga obavlja grananje programa. Za bilo koji dani

izraz moguće je reći kako može biti istinit ili lažan. Primjerice, izjavom da na kišoviti dan nosimo kišobran, za zadani dan ispituje se meteorološko vrijeme i temeljem rezultata tog ispitivanja donosi se odluka o nošenju ili ne nošenju



kišobrana. Generalizirano odlučivanje prikazano je na primjeru slike 2.7.



Sl. 2.7. *Odlučivanje*

U Pythonu odlučivanje moguće je pisati na sljedeći način:

```
kisovito_vrijeme = True  
if ( kisovito_vrijeme == True):  
    print "Danas pada kisa. Ponesite kisobran."
```

2.4. Petlje

Petlje su apsolutno neophodne u programskim jezicima, i one omogućuju višestruko izvođenje programskog koda. Generalno, naredbe se izvode sekvencijalno i ponekad je potrebno jedan dio programskog koda izvršiti nekoliko puta. Primjerice, zadan je zadatak ispisa brojeva od 1 do 1000. Jedan, iako apsolutno redundantan način za obavljanje ove operacije bi bio da programer ručno unese 1000 brojeva. Vrijeme koje bi bilo potrebno za ručni unos 1000 brojeva je neprihvatljiv za bilo kakavo ozbiljno programiranje, stoga se problem rješava pomoću petlji na način prikazan na slici 2.8. Važno je napomenuti da postoji više načina za rješavanje ovog problema jer postoje različite vrste petlji .

```
1 for broj in range(1,1001):
2     print broj
```

```
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
valsymoth@debian:~/Python$
```

Sl. 2.8. Ispis brojeva od 1 do 1000

2.4.1. For petlja

Dijagram toka for petlje prikazan je na slici 2.9, a na slici 2.10. prikazan je programski kod i rezultat izvršavanja.

Sl. 2.9. *Dijagram toka for petlje*


```
1 for slovo in "Python":  
2     print "Trenutno slovo: ", slovo
```

```
valsymoth@debian: ~/Python/lp$ python petlje.py  
Trenutno slovo: P  
Trenutno slovo: y  
Trenutno slovo: t  
Trenutno slovo: h  
Trenutno slovo: o  
Trenutno slovo: n  
valsymoth@debian: ~/Python/lp$ 
```

Sl. 2.10. Primjer for petlje

Na slici 2.11. prikazan je programski kod i rezultat izvršavanja indeksiranog ispisa for petlje.

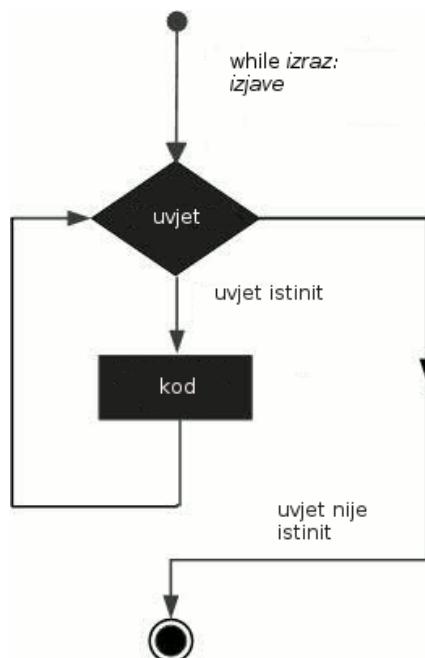
```
1 automobili = [ "BMW", "Tesla", "Concept_One" ]
2
3 for index in range(len(automobili)):
4     print "Trenutni automobil: ", automobili[index]
5
```

valsymoth@debian: ~/Python/lp\$ python petlje.py
Trenutni automobil: BMW
Trenutni automobil: Tesla
Trenutni automobil: Concept_One
valsymoth@debian: ~/Python/lp\$ █

Sl. 2.11. Primjer indeksiranog ispisa for petlje

2.4.2. While petlja

while petlja razlikuje se od for petlje jer će se ona izvršavati sve dok je uvjet istinit i dijagram toka prikazan je na slici 2.12.



Sl. 2.12. Dijagram toka while petlje

Na slici 2.13. vidljiv je programski kod i rezultat izvršavanja `while` petlje.

```
1 brojac = 0
2 while brojac < 5:
3     print "Brojac iznosi: ", brojac
4     brojac += 1 # jako važno!
```

```
valsymoth@debian: ~/Python/lp$ python petlje.py
Brojac iznosi: 0
Brojac iznosi: 1
Brojac iznosi: 2
Brojac iznosi: 3
Brojac iznosi: 4
valsymoth@debian: ~/Python/lp$
```

Sl. 2.13. Primjer `while` petlje

Nužno je naglasiti kako je kod `while` petlje važno modificirati uvjet na takav način da se spriječi javljanje beskonačne petlje. Beskonačna petlja nastaje kada ispitivanje `while` uvjeta konstantno rezultira u logičkoj jedinici.

```
1 # beskonacna petlja
2
3 while True:
4     broj = raw_input("Unesite broj: ")
5     print broj
6 print "Ja se nikada necu ispisati"
```

```
valsymoth@debian: ~/Python/lp$ python petlje.py
Unesite broj: 1
1
Unesite broj: 2
2
Unesite broj: 3
3
Unesite broj: a
a
Unesite broj: -33
-33
Unesite broj: ^CTraceback (most recent call last):
  File "petlje.py", line 4, in <module>
    broj = raw_input("Unesite broj: ")
KeyboardInterrupt
valsymoth@debian: ~/Python/lp$
```

Sl. 2.14. Primjer beskonačne `while` petlje

Posljednja linija programskog koda u navedenom primjeru nikada se neće izvoditi jer program ne izlazi iz `while` petlje. Kako bi izašli iz takvog programa nužno ga je prekinuti, specifično gledajući u Linuxima pritiskom tipke **CTRL + C**. Pritiskom tipke **CTRL + C** operacijski sustav šalje prekid i time zaustavlja rad programa.

2.5. Funkcije

Funkcije su blokovi organiziranog, ponovno iskoristivnog, koda koji ima zadatak obavljanje nekog složenijeg zadatka. Funkcije pružaju aplikacijama modularnost jer se funkcije mogu više puta iskoristiti unutar istog programa. Postoje ugrađene funkcije poput `raw_input()`, te korisničke funkcije.

2.5.1. Definiranje funkcija

Svaka funkcija u programskom jeziku Python mora započinjati ključnom riječi `def` nakon čega slijedi naziv funkcije te zatvorene zagrade tj. `()`. Funkcije mogu primati vrijednosti, stoga ukoliko funkcija prima neke vrijednosti one se navode unutar zagrade. Nakon zagrade obavezno slijedi znak dvotočja. Prva izjava nije nužna, te može poslužiti kao dokumentacija. Svaka sljedeća izjava mora biti ispravno uvučena te ići jedna ispod druge. Funkcija završava u trenutku kada razina identacije završi na nuli.

```
def ime_funkcije ( parametri ):
    "Dokumentacija funkcije"
    izjava 1
    izjava 2
    ...
    return [ izraz ]
```

2.5.2. Funkcijski poziv

Definicijom je funkciji dodijeljen naziv, parametri koje prima, kod koji se izvršava te povratni izraz. Kako bi se ta funkcija izvršila nužno ju je pozvati, a poziva se na način da se upiše naziv funkcije te da se navedu potrebni parametri. Na slici 2.15. možemo vidjeti primjer funkcijskog poziva.

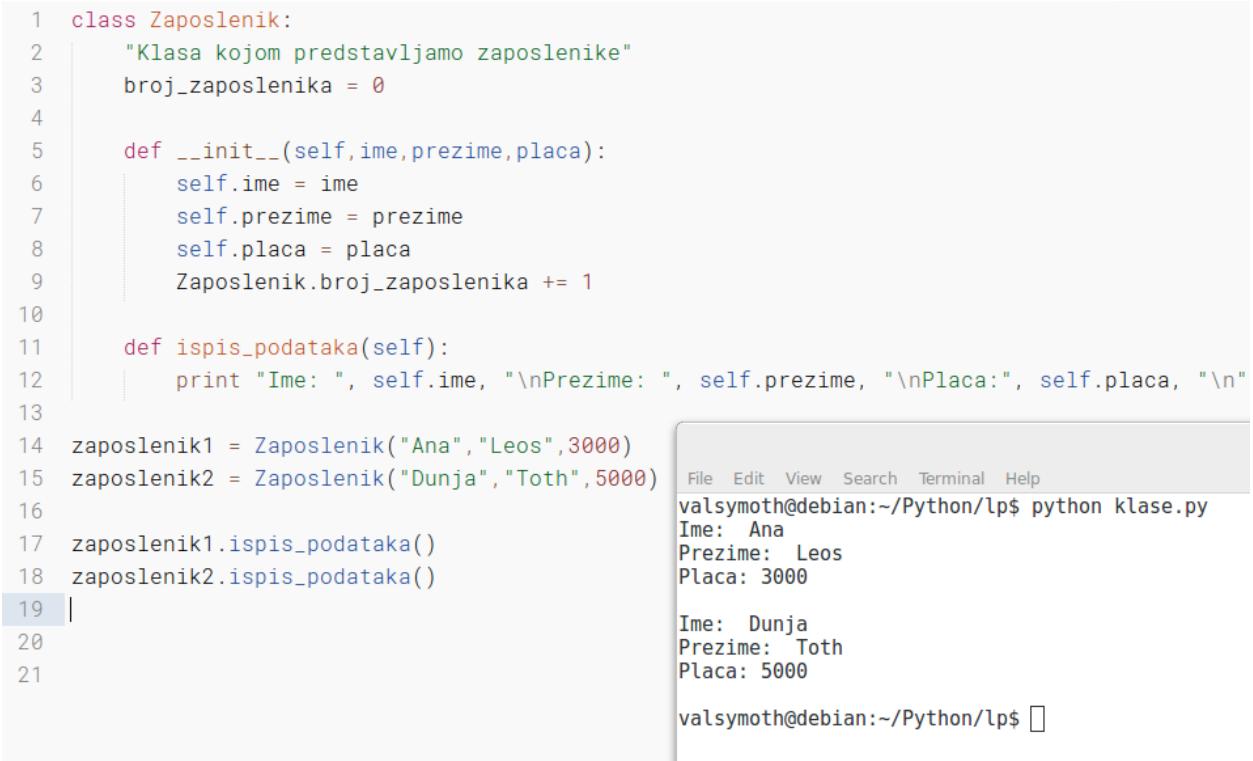


```
1 def zbroji(a,b):
2     "Funkcija zbraja dva broja"
3     return a + b
4
5 print zbroji(10,30)
6 print zbroji(32.4,55.7)
7 print zbroji(-351.315,505)
8
9 File Edit View Search Terminal Help
10 valsymoth@debian:~/Python/lp$ python funkcije.py
11 40
12 88.1
13 153.685
14 valsymoth@debian:~/Python/lp$ 
```

Sl. 2.15. Primjer funkcijskog poziva

2.6. Klase i objekti

U objektno orijentiranim programskim jezicima klasom se nazivaju dijelovi programskog koda koji služe kao predlošci za stvaranje korisničkih tipova podataka. Objekt je instanca određene klase čije je ponašanje i inicijalno stanje definirano unutar klase. Na slici 2.16. prikazan je programski kod stvaranja klase i objekta, te rezultat izvršavanja.



```
1 class Zaposlenik:
2     "Klasa kojom predstavljamo zaposlenike"
3     broj_zaposlenika = 0
4
5     def __init__(self,ime,prezime,placa):
6         self.ime = ime
7         self.prezime = prezime
8         self.placa = placa
9         Zaposlenik.broj_zaposlenika += 1
10
11    def ispis_podataka(self):
12        print "Ime: ", self.ime, "\nPrezime: ", self.prezime, "\nPlaca:", self.placa, "\n"
13
14 zaposlenik1 = Zaposlenik("Ana","Leos",3000)
15 zaposlenik2 = Zaposlenik("Dunja","Toth",5000)
16
17 zaposlenik1.ispis_podataka()
18 zaposlenik2.ispis_podataka()
19
20
21
```

valsymoth@debian:~/Python/lp\$ python klase.py
Ime: Ana
Prezime: Leos
Placa: 3000

Ime: Dunja
Prezime: Toth
Placa: 5000
valsymoth@debian:~/Python/lp\$

Sl. 2.16. Primjer klase i objekta

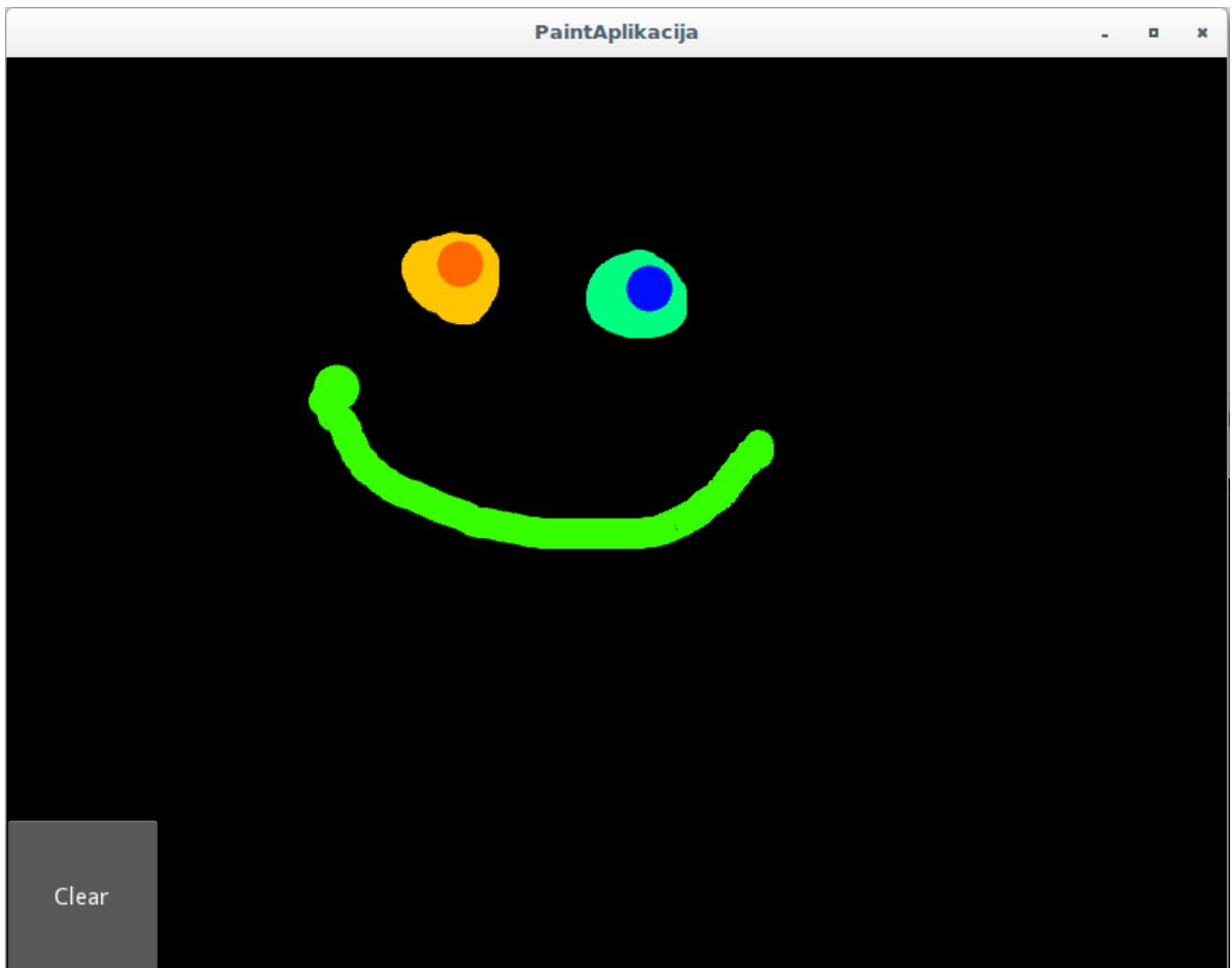
Varijabla `broj_zaposlenika` je klasna varijabla čiju vrijednost dijeli svi objekti klase `Zaposlenik`. Prva metoda `__init__()` je posebna metoda koja se naziva konstruktor ili inicijalizacijska metoda koja se poziva prilikom stvaranja novog objekta. Sve ostale metode unutar klase pišu se poput normalnih funkcija. Kako bi se stvorila instanca klase, tj. objekt, nužno je koristiti njegov naziv uz listu argumenata, kao što je u primjeru vidljivo: `Zaposlenik("Ana", "Leos", 3000)`.

3. PRIMJERI APLIKACIJA U PROGRAMSKOM JEZIKU PYTHON

U ovom poglavlju biti će prikazane aplikacije napisane u programskom jeziku Python koristeći Kivy framework. Kivy je besplatano (MIT licenca), više-platformska (Linux, Windows, OS X, Android, iOS) rješenje za izradu grafičkih sučelja i odabran je zbog svoje jednostavnosti.

3.1. Kivy aplikacija za crtanje

Cilj ove jednostavne aplikacije jest omogućiti korisniku crtanje jednostavnijih crteža, uz mogućnost brisanje slike. Također bit će jasno vidljivo koliko malo linija koda je potrebno kako bi se napisala funkcionalna Kivy aplikacija. Programski kod biti će prikazan segmentno uz objašnjenje pojedinih dijelova, nakon čega će biti prikazan kod u cijelosti. Slika 3.1. prikazuje konačan izgled aplikacije za crtanje.



Slika 3.1 *Paint aplikacija*

Aplikacija započinje uvođenjem pomoćnih biblioteka i funkcija koje omogućavaju daljni rad:

```
from random import random
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.uix.button import Button
from kivy.graphics import Color, Ellipse, Line
```

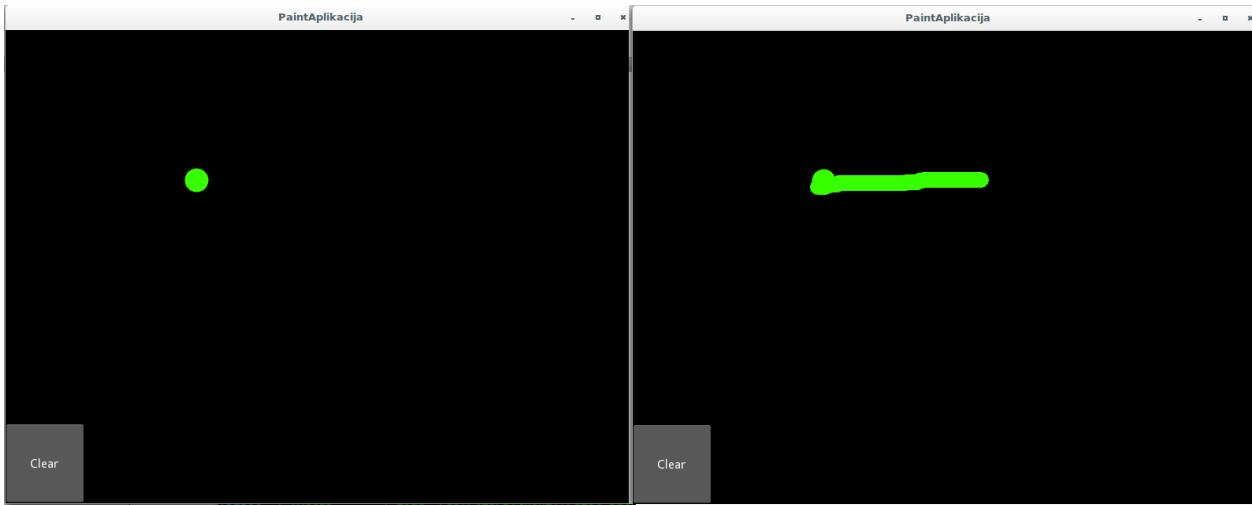
Svaka Kivy aplikacija može se gledati kao stablo **Widget** elemenata. Svaki **widget** element obuhvaća podatke, definira interakciju korisnika i ispisuje grafičke podatke. U našem slučaju definirali smo vlastiti **widget** naziva “MyPaintWidget” i pripadajuću korisničku interakciju:

```
class MyPaintWidget(Widget):

    def on_touch_down(self, touch):
        color = (random(), 1, 1)
        with self.canvas:
            Color(*color, mode=' hsv ')
            d = 30.
            Ellipse(pos=(touch.x - d / 4, touch.y - d / 4), size=(d, d))
            touch.ud[ 'line' ] = Line(points=(touch.x, touch.y), width = 10)

    def on_touch_move(self, touch):
        touch.ud[ 'line' ].points += [touch.x, touch.y]
```

Dvije metode navedene ovdje su `on_touch_down` i `on_touch_move`. Metoda `on_touch_down` definira što se događa u trenutku kada korisnik započne interakciju s aplikacijom što može biti mišem ili dodirom. U primjeru navedenom gore, definira se nasumična RGB vrijednost kao vrijednost boje i ispisuje se točka. U trenutku korisnikovog pomicanja dodirne točke, vrijednosti boje se upisuju na odgovarajuće koordinate i na taj način je dobivena linija. Na slici 3.2. prikazana je prvotna točka i linija u trenutku pomicanja.



Slika 3.2 *Paint aplikacija - točka i linija*

Sljedeću definirana klasa je `PaintAplikacijaApp`. Ovaj naziv je redundantan, no prilikom pokretanja aplikacije, Kivy koristi naziv prije App dijela kao naziv aplikacije. Unutar klase definiraju se `build` i `clear_canvas` metode. Unutar `build` metode definirana je jezgra aplikacije, poziv grafičkih elemenata te vraćanje podataka u grafičkom obliku. `clear_canvas` metoda omogućuje brisanje prethodnog crteža:

```
class PaintAplikacijaApp(App):

    def build(self):
        parent = Widget()
        self.painter = MyPaintWidget()
        clearbtn = Button(text='Clear')
        clearbtn.bind(on_release=self.clear_canvas)
        parent.add_widget(self.painter)
        parent.add_widget(clearbtn)
        return parent

    def clear_canvas(self, obj):
        self.painter.canvas.clear()
```

Naš program završava zaštitom od kolizije prilikom import funkcionalnosti i pokreće aplikaciju na sljedeći način:

```
if __name__ == '__main__':
    PaintAplikacijaApp().run()
```

3.2. Kivy kalkulator

Kao sljedeći primjer prikazujemo aplikaciju kalkulatora. U ovom primjeru prikazano je odvajanje prezentacijskog od aplikacijskog sloja što je baza svih naprednijih aplikacija koje se mogu napisati koristeći Kivy i Python. Programski kod biti će prikazan segmentno uz objašnjenje pojedinih dijelova. Na slici 3.3. prikazana je aplikacija u izvođenju.



Slika 3.3 *Kalkulator*

Aplikacija započinje uvođenjem pomoćnih biblioteka i funkcija koje nam omogućuju daljni rad:

```
import kivy
kivy.require('1.8.0')
from kivy.app import App
from kivy.uix.gridlayout import GridLayout
```

GridLayout omogućava korištenje responzivne grafičke matrice s kojom je definirano samo sučelje aplikacije. Klasa `CalcGridLayout(GridLayout)` predstavlja grafičko sučelje i u njoj su definirane metode koje se koriste unutar aplikacije.

```
class CalcGridLayout(GridLayout):

    def calculate(self,calculation):
        if calculation:
            try:
                self.display.text = str(eval(calculation))
            except Exception:
                self.display.text = "Error"
```

Metoda `calculate` izračunava konačni izraz koji je spremljen unutar tekstualnog polja. Izraz `str(eval(calculation))` koristi se radi evaluacije numeričkog izraza i pretvorbu u podatak tipa `string`. Rezultat je prikazan unutar glavnog polja za ispis rezultata. Sljedeći korak je definiranje same aplikacije i njezino pokretanje:

```
class CalculatorApp(App):
    def build(self):
        return CalcGridLayout()

calcApp = CalculatorApp()
calcApp.run()
```

Cijelu aplikaciju pohranjena je kao datoteku naziva `main.py`. Prvi dio aplikacije uspješno je napisan. Sljedeći korak je napisati datoteku koja će definirati grafičko sučelje same aplikacije. U Kivyu grafičkoj datoteci dodjelimo ekstenziju `.kv`. Na taj način se povezuje Python aplikacija s grafičkim sučeljem definiranim u `.kv` datoteci. U primjeru prve aplikacije to nije učinjeno obzirom da je aplikacija jednostavna, no bilo koja kompleksnija aplikacija mora slijediti ovaj način odvajanja aplikacijskog od prezentacijskog sloja jer bi u protivnom bila narušena čitljivost aplikacijskog koda.

```
1 # Definiranje korisnicke tipke
2 <CustButton@Button>:
3     font_size: 32
4
5 #Dodata naziva(id) za lakse referenciranje
6 <CalcGridLayout>:
7     id: calculator
8     display: entry
9     rows: 5
10    spacing: 10
11    padding: 10
12
13
14 # Mjesto na kojem se rezultati ispisuju
15 BoxLayout:
16     TextInput:
17         id: entry
18         multiline: False
19         font_size: 32
20
21
22 # kada je tipka pritisnuta, promjeni vrijednost
23 BoxLayout:
24     spacing: 10
25     CustButton:
26         text: "7"
27         on_press: entry.text += self.text
28     CustButton:
29         text: "8"
30         on_press: entry.text += self.text
31     CustButton:
```

Slika 3.4 *calculator.kv* prvi dio koda

```
31     CustButton:
32         text: "9"
33         on_press: entry.text += self.text
34     CustButton:
35         text: "+"
36         on_press: entry.text += self.text
37
38     BoxLayout:
39         spacing: 10
40         CustButton:
41             text: "4"
42             on_press: entry.text += self.text
43         CustButton:
44             text: "5"
45             on_press: entry.text += self.text
46         CustButton:
47             text: "6"
48             on_press: entry.text += self.text
49         CustButton:
50             text: "-"
51             on_press: entry.text += self.text
52
53     BoxLayout:
54         spacing: 10
55         CustButton:
56             text: "1"
57             on_press: entry.text += self.text
58         CustButton:
59             text: "2"
60             on_press: entry.text += self.text
```

Slika 3.5 calculator.kv drugi dio koda

```

61     CustButton:
62         text: "3"
63         on_press: entry.text += self.text
64     CustButton:
65         text: "*"
66         on_press: entry.text += self.text
67
68 # poziv metode/funkcije u trenutku pritiska znaka "="
69 BoxLayout:
70     spacing: 10
71     CustButton:
72         text: "AC"
73         on_press: entry.text = ""
74     CustButton:
75         text: "0"
76         on_press: entry.text += self.text
77     CustButton:
78         text: "="
79         on_press: calculator.calculate(entry.text)
80     CustButton:
81         text: "/"
82         on_press: entry.text += self.text

```

Slika 3.6 *calculator.kv* treći dio koda

3.3 Distribucija i stvaranje .deb datoteke

Deb je format programskih datoteka na Debian Linux distribuciji. Debian paketi su standardne Unix ar datoteke koje sadrže dvije tar arhive. Jedna sadrži kontrolne informacije, dok druga sadrži podatke koje se instaliraju. **Dpkg** (Debian Package Manager) je alat pomoću kojeg se obavlja instalacija .deb datoteka iako krajnji korisnici najčešće nemaju interakciju s dpkg već s APT menadžerom paketa poput Synaptic-a.

Postupak stvaranja .deb datoteke:

Tablica 3.1 *Koraci stvaranja .deb datoteke*

1. korak	Otvaranje terminala i stvaranje datoteke pomoću naredbe: “mkdir kalkulator_1.0-1”
2. korak	Stvaranje kontrolnih direktorija pomoću naredbi: “mkdir kalkulator_1.0-1/usr” “mkdir kalkulator_1.0-1/usr/local” “mkdir kalkulator_1.0-1/usr/local/bin”
3. korak	Kopiranje programa u direktoriji “ kalkulator_1.01/usr/local/bin”
4. korak	Stvaranje “Debian” datoteke pomoću naredbe: “mkdir kalkulator_1.0-1/DEBIAN”
5. korak	Stvaranje “control” datoteke upisujući naredbu: “vim kalkulator_1.0-1/DEBIAN/control” i upisujemo unutar datoteke: Package: Kalkulator Version: 1.01 Section: base Priority: optional Architecture: i386 Depends: nekaBiblioteka (>= 1.2.13), sljedećaBiblioteka(>= 1.2.6) Maintainer: Vaše Ime <you@email.com> Description: Kalkulator
6. korak	Pakiranje u “.deb” datoteku pomoću naredbe: “dpkg-deb --build helloworld_1.01”

4. ZAKLJUČAK

Moderni programski jezici današnjice moraju efikasno stvarati aplikacije sutrašnjice. Zahtjevi za kvalitetom, optimalnim radom, skalabilnošću i sigurnošću modernih programa sve su veći. Suvremeni programer danas mora na takve zahtjeve odgovarati odabirom skupa alata koji mu neće stajati na putu ostvarivanja tih zadataka. Python predstavlja jednu od mogućih opcija koje programer može uključiti u svoj razvojni proces. Kvaliteta Python koda je konceptualno integrirana u sam dizajn jezika i time se ostvaruje optimalni rad, jer se s manjim opsegom programskog koda može postići puno više nego u klasičnim jezicima “niže razine”. Sigurnost danas je važnija nego ikada prije, no razvojnim procesom jezika koji se temelji na “open source” principima to se uistinu lako postiže. Od jednostavnih početničkih aplikacija do znanstveno-istraživačkih kalkulacija, Python svojom skalabilnošću pogoduje svima. Guido van Rossum imao je veliku viziju stvaranja programskog jezika koji će predstavljati budućnost, i ta vizija je danas zasigurno ostvarena. Python je programski jezik koji će se nastaviti razvijati i utjecati na buduće programere na nezamislive načine, a budućnost mu se čini sve sjajnijom obzirom na nadolazeće potrebe “Internet of Things” i “Big Data” aplikacija.

5. LITERATURA

- [1] Z. A. Shaw, Learn Python The Hard Way, Addison - Wesley Professional, 11. listopad 2013.
- [2] M. Lutz, Learning Python, O'Reilly Media, 6. srpanj 2013.
- [3] A. Sweigart, Automate the Boring Stuff with Python: Practical Programming for Total Beginners, No Starch Press, 1. svibanj 2015
- [4] L. Ramalho, Fluent Python: Clear, Concise, and Effective Programming, O'Reilly Media, 20. kolovoz 2015.
- [5] E. Matthes Python Crash Course: A Hands-On, Project-Based Introduction to Programming, No Starch Press, 30. studeni 2015.
- [6] Sužbena stranica programskog jezika Python [Mrežno]. Dostupno:
<https://www.python.org/> [Pokušaj pristupa 22 9 2016]
- [7] Kivy službena stranica i dokumentacija [Mrežno]. Dostupno:
<https://kivy.org/docs/api-kivy.html> [Pokušaj pristupa 22 9 2016]
- [8] Tutorialpoint pregled Python programskog jezika. [Mrežno]. Dostupno:
<http://www.tutorialspoint.com/python/> [Pokušaj pristupa 22 9 2016]
- [9] Stackoverflow pitanja i odgovori u vezi programskog jezika Python. [Mrežno]. Dostupno:
<http://stackoverflow.com/questions/tagged/python> [Pokušaj pristupa 22 9 2016]
- [10] Codeschool pregled Python programskog jezika. [Mrežno]. Dostupno:
<https://www.codeschool.com/learn/python> [Pokušaj pristupa 22 9 2016]
- [11] CodeAcademy pregled Python programskog jezika. [Mrežno.] Dostupno:
<https://www.codecademy.com/learn/python> [Pokušaj pristupa 22 9 2016]
- [12] Sololearn pregled Python programskog jezika. [Mrežno.] Dostupno:
<https://www.sololearn.com/Course/Python/> [Pokušaj pristupa 22 9 2016]
- [13] LearnXinYminutes pregled Python programskog jezika. [Mrežno.] Dostupno:
<https://learnxinyminutes.com/docs/python/> [Pokušaj pristupa 22 9 2016]
- [14] CodingBat pregled Python programskog jezika. [Mrežno.] Dostupno:
<http://codingbat.com/python> [Pokušaj pristupa 22 9 2016]
- [15] PlanetPython pregled Python programskog jezika. [Mrežno.] Dostupno:
<http://planetpython.org/> [Pokušaj pristupa 22 9 2016]

SAŽETAK

Naslov: PROGRAMSKI JEZIK PYTHON

Ideja programskog jezika Python nastaje u kasnim osamdesetim godinama od strane nizozemskog programera Guido van Rossuma koji je imao viziju stvaranja jezika koji će imati mogućnosti rukovanja iznimkama i služiti kao moguće sučelje Amoeba operacijskom sustavu. U veljači 1991. godine van Rossum objavljuje prvu verziju programskog jezika (verzija 0.9.0) s podrškom za objektno orijentirano programiranje i ugrađenim tipovima podataka poput list, dict, str i drugim. 16. 8. 2000. godine objavljuje se verzija 2.0. s brojnim novim dodacima od kojih su značajni ciklično detektirajući upravljač memorijom te podrška za Unicode znakovni standard.

Ključne riječi: Python, Guido van Rossum, programiranje Python, Kivy, sintaksa

ABSTRACT

Title: PROGRAMMING LANGUAGE PYTHON

The idea for Python programming language was conceived in late 1980's and its implementation began in December 1989 by a Dutch programmer Guido van Rossum in Netherlands. Python is a programming language developed with a purpose of improving programmers productivity, by means of portability, modularity and ease of use. Quality is achieved with Python's easy to read syntax, productivity is improved by reducing the lines of code necessary to write to achieve a specific goal. Portability and modularity are result of Python's open source implementation and a vast number of libraries that extend Python language.

Keywords: Python, programming language, Guido van Rossum, Python, Kivy syntax

ŽIVOTOPIS

Tomislav Šanitć, rođen 21. 4. 1993. u Vinkovcima. Pohađao osnovnu školu "Antun Gustav Matoš" nakon čega upisuje ekonomsku i trgovacku školu Ivana Domca, također u Vinkovcima. Po završetku srednješkolskog obrazovanja upisuje Elektrotehnički fakultet u Osijeku. Uz studiji bavi se izradom i dizajnom web stranica.

PRILOG A. Izvorni kod aplikacije za crtanje

```
from random import random
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.uix.button import Button
from kivy.graphics import Color, Ellipse, Line
class MyPaintWidget(Widget):

    def on_touch_down(self, touch):
        color = (random(), 1, 1)
        with self.canvas:
            Color(*color, mode=' hsv ')
            d = 30.
            Ellipse(pos=(touch.x - d / 4, touch.y - d / 4), size=(d, d))
            touch.ud['line'] = Line(points=(touch.x, touch.y), width = 10)

    def on_touch_move(self, touch):
        touch.ud['line'].points += [touch.x, touch.y]

class PaintApplikacijaApp(App):

    def build(self):
        parent = Widget()
        self.painter = MyPaintWidget()
        clearbtn = Button(text='Clear')
        clearbtn.bind(on_release=self.clear_canvas)
        parent.add_widget(self.painter)
        parent.add_widget(clearbtn)
        return parent

    def clear_canvas(self, obj):
        self.painter.canvas.clear()
if __name__ == '__main__':
    PaintApplikacijaApp().run()
```