/VN1Z 6TFS (VJE

3FMFBTF

8SJUUFO CZ UIF /VN1Z DPNN

+VOF

```
8IBU JT /VN1Z
/VN1Z RVJDLTUBSU
/VN1Z  UIF BCTPMVUF CBTJDT GPS CFHJOOFST
/VN1Z GVOEBNFOUBMT
.JTDFMMBOFPVT
/VN1Z GPS ."5-"# VTFST
#VJMEJOH GSPN TPVSDF
6TJOH /VN1Z $ "1*
/VN1Z )PX 5PT
'PS EPXOTUSFBN QBDLBHF BVUIPST
' 1: VTFS HVJEF BOE SFGFSFODF NBOVBM
(MPTTBSZ
6OEFS UIF IPPE %PDVNFOUBUJPO GPS EFWFMPQFST
3FQPSUJOH CVHT
3FMFBTF OPUFT
/VN1Z MJDFOTF
1ZUIPO .PEVMF *OEFY
*OEFY
```

6JKU IWKFG KU CP QXGTXKGY CPF GZRNCKPU VJG KORQTVCPV HGCVWTGU F

8)"5 *4 /6.1:

0WO2[KU VJG HWPFCOGPVCN RCEMCIG HQT UEKGPVK·E EQORWVKPI KP 2[VJ
OGPUKQPCN CTTC[QDLGEV XCTKQWU FGTKXGF QDLGEVU UWEJ CU OCUMGF
HCUV QRGTCVKQPU QP CTTC[U KPENWFKPI OCVJGOCVKECN NQIKECN UJC
VTCPUHQTOU DCUKE NKPGCT CNIGDTC DCUKE UVCVKUVKECN QRGTCVKQPU

#V VJG EQTG QH VJG 0WO2[RCEMCIG KU VJG PFCTTC[QDLGEV 6JKU GPECRUWNCVGU CTTC[U QH JQOQ
V[RGU YKVJ OCP[QRGTCVKQPU DGKPI RGTHQTOGF KP EQORKNGF EQFG HQT
DGVYGGP 0WO2[CTTC[U CPF VJG UVCPFCTF 2[VJQP UGSWGPEGU

Y0WO2[CTTC[U JCXG C ·ZGF UK\G CV ETGCVKQP WPNKMG 2[VJQP NKUVU
QH OPB SYKBZN ETGCVG C PGY CTTC[CPF FGNGVG VJG QTKIKPCN

Y6JG GNGOGPVU KP C 0WO2[CTTC[CTG CNN TGSWKTGF VQ DG QH VJG UC
OGOQT[6JG GZEGRVKQP QPG ECP JCXG CTTC[U QH 2[VJQP KPENWFKP
QH FK¶GTGPV UK\GF GNGOGPVU

Y0WO2[CTTC[U HCEKNKVCVG CFXCPEGF OCVJGOCVKECN CPF QVJGT V[RGU
UWEJ QRGTCVKQPU CTG GZGEWVGF OQTG G¹EKGPVN[CPF YKVJ NGUU EQF

Y# ITQYKPI RNGVJQTC QH UEKGPVK·E CPF OCVJGOCVKECN 2[VJQP DCUGF
V[RKECNN[UWRRQTV 2[VJQP UGSWGPEG KPRWV VJG[EQPXGTV UWEJ K
QHVGP QWVRWV 0WO2[CTTC[U +P QVJGT YQTFU KP QTFGT VQ G¹EKGPVN
VK·E OCVJGOCVKECN 2[VJQP DCUGF UQHVYCTG LWUV MPQYKPI JQY VQ
QPG CNUQ PGGFU VQ MPQY JQY VQ WUG 0WO2[CTTC[U

6JG RQKPVU CDQWV UGSWGPEG UK\G CPF URGGF CTG RCTVKEWNCTN[KORQT
VJG ECUG QH OWNVKRN[KPI GCEJ GNGOGPV KP C & UGSWGPEG YKVJ VJG EQ
NGPIVJ +H VJG FCVC CTG UVQTGF KP C 2[VJQP NKUV C YG EQWNF NQQR QXGT GCEJ GNGOGPV

D <>
GPSJ JOSBOHMFOB
DBQQFOE BC<J>

6JKU RTQFWEGU VJG EQTTGEV CPUYGT DWV KH C CPF D GCEJ EQPVCKP OKNNKQPU QH PWODGTU YG YKN
·EKGPEKGU QH NQQRKPI KP 2[VJQP 9G EQWNF CEEQORNKUJ VJG UCOG VCUM
PGINGEV XCTKCDNG FGENCTCVKQPU CPF KPKVKCNK\CVKQPU OGOQT[CNNQ

GPSJ J SPXT J \
D<J> B<J>C<J>
^

6JKU UCXGU CNN VJG QXGTJGCF KPXQNXGF KP KPVGTRTGVKPI VJG 2[VJQP EQ
QH VJG DGPG·VU ICKPGF HTQO EQFKPI KP 2[VJQP (WTVJGTOQTG VJG EQFKP
QH QWT FCVC +P VJG ECUG QH C & CTTC[HQT GZCORNG VJG % EQFG CDT

```
GPS J      J   SPXT  J \
  GPS K      K   DPMVNOT  K
    D<J><K>B<J><K>C<J><K>
  ^
^
```

0WO2[IKXGU WU VJG DGUV QH DQVJ YQTNFU GNGOGPV D[GNGOGPV QRGT
KPXQNXGF DWV VJG GNGOGPV D[GNGOGPV QRGTCVKQP KU URGGFKN[GZGE

```
D  B  C
```

FQGU YJCV VJG GCTNKGT GZCORNGU FQ CV PGCT % URGGFU DWV YKVJ VJG
2[VJQP +PFGGF VJG 0WO2[KFKQO KU GXGP UKORNGT 6JKU NCUV GZCORN
DCUKU QH OWEJ QH KVU RQYGT XGEVQTK\CVKQP CPF DTQCFECUVKPI

8IZ JT /VN1Z 'BTU

8GEVQTK\CVKQP FGUETKDGU VJG CDUGPEG QH CP[GZRNKEKV NQQRKPI KPF
EQWTUG LWUV SDGJKPF VJG UEGPGUT KP QRVKOK\GF RTG EQORKNGF % EQ
CTG

YXGEVQTK\GF EQFG KU OQTG EQPEKUG CPF GCUKGT VQ TGCF

YHGYGT NKPGU QH EQFG IGPGTCNN[OGCPU HGYGT DWIU

YVJG EQFG OQTG ENQUGN[TGUGODNGU UVCPFCTF OCVJGOCVKECN PQVC
  OCVJGOCVKECN EQPUVTWEVU

YXGEVQTK\CVKQP TGUWNVU KP OQTG S2[VJQPKET EQFG 9KVJQWV XGEVQ
  CPF FK¹EWNV VQ TGCF

$TQCFECUVKPI KU VJG VGTO WUGF VQ FGUETKDG VJG KORNKEKV GNGOGPV
KP 0WO2[CNN QRGTCVKQPU PQV LWUV CTKVJOGVKE QRGTCVKQPU DWV NQ
GNGOGPV D[GNGOGPV HCUJKQP K G VJG[DTQCFECUV
CTTC[U QH VJG UCOG UJCRG QT C UECNCT CPF CP CTTC[QT GXGP VYQ CTTC
CTTC[KU SGZRCPFCDNGT VQ VJG UJCRG QH VJG NCTIGT KP UWEJ C YC[VJCV
STWNGUT QH DTQCFECUVKPI UGG

8IP &MTF 6TFT /VN1Z

0WO2[HWNN[UWRRQTVU CP QDLGEV QTKGPVGF CRRTQCEJ UVCTVKPI QPE
RQUUGUUKPI PWOGTQWU OGVJQFU CPF CVVTKDWVGU /CP[QH KVU OGVJQF
PCOGURCEG CNNQYKPI VJG RTQITCOOGT VQ EQFG KP YJKEJGXGT RCTCFKIO
CTTC[FKCNGEV VQ EQFG QH OWNVK FKOGPUKQPCN FCV
2[VJQP

/6.1: 26*$,45"35

1SFSFRVJTJUFT

;QWPNN PGGF VQ MPQY C DKV QH 2[VJQP VQ IGV VJG OQUV QWV QH VJKU EJCRVGT. UGG VJG

6Q YQTM VJG GZCORNGU KP VJKU EJCRVGT, [QW'NN PGGF KP CFFKVKQP VQ 0WO2[

-FBSOFS QSPƒMF

6JKU KU C SWKEM QXGTXKGY QH CTTC[U KP 0WO2[... ECP DG OCPKRWNCVGF. +P RCTVKEWNCT, KH [QW FQPPV MPQY JQY VQ CRRN[... WUKPI HQT NQQRU, QT KH [QW YCPV VQ WPFGTUVCPF CZKU CPF UJCRG RTQRGTVKGU ... JGNR.

-FBSOJOH 0CKFDUJWFT

#HVGT TGCFKPI [QW UJQWNF DG CDNG VQ:

- Y7PFGTUVCPF VJG FK¶GTGPEG DGVYGGP QPG-, VYQ-, CPF P-FKOGPUKQPCN ...
- Y7PFGTUVCPF JQY VQ CRRN[UQOG NKPGCT CNIGDTC QRGTCVKQPU VQ P-F ...
- Y7PFGTUVCPF CZKU CPF UJCRG RTQRGTVKGU HQT P-FKOGPUKQPCN CTTC[...

5IF #BTJDT

0WO2[PU OCKP QDLGEV KU VJG JQOQIGPGQWU OWNVKFKOGPUKQPCN CTTC[. ... UCOG V[RG, KPFGZGF D[C VWRNG QH PQP-PGICVKXG KPVGIGTU. +P 0WO2[FK...

(QT GZCORNG, VJG CTTC[HQT VJG EQQTFKPCVGU QH C RQKPV KP 3& URCEG ... KP KV, UQ YG UC[KV JCU C NGPIVJ QH 3. +P VJG GZCORNG RKEVWTGF DGNQY, VJG UGEQPF CZKU JCU C NGPIVJ QH 3.

<< >
< >>

0WO2[PU CTTC[ENCUU KU ECNNGF ndarray. ... VJG UCOG CU VJG 5VCPFCTF 2[VJQP .KDTCT[ENCUU array.array, YJKEJ QPN[JCPFNGU QPG-FKOGPUKQPCN ... HWPEVKQPCNKV[. 6JG OQTG KORQTVCPV CVVTKDWVGU QH CP ndarray QDLGEV ...

OEBSSBZ OEJN

VJG PWODGT QH CZGU (FKOGPUKQPU) QH VJG CTTC[.

OEBSSBZ TIBQF

VJG FKOGPUKQPU QH VJG CTTC[6JKU KU C VWRNG QH KPVGIGTU KPFKE
OCVTKØTYQKYUJCEFNWTOBQYFKNNODNG 6JG NGPIVJBQHFWJRGNGKU VJGTGHQTG VJ
QH CØEUN

OEBSSBZ TJ[F

VJG VQVCN PWODGT QH GNGOGPVU QH VJG CTTC[TBKQF KU GSWCN VQ VJ

OEBSSBZ EUZQF

CP QDLGEV FGUETKDKPI VJG V[RG QH VJG GNGOGPVU KP VJG CTTC[1PG
V[RGU #FFKVKQPCNN[0WO2[RTQXKFGU V[RGU QH KVU QYP PWOR[KPV
GZCORNGU

OEBSSBZ JUFNTJ[F

VJG UK\G KP D[VGU QH GCEJ GNGOGPV QH VJG CTTC[CP CT
YJKNG QØ NQH MFJRCUU FNTJ[F +V KU GSWEKXCNGPV VQ
JUFNTJ[F

OEBSSBZ EBUB

VJG DW¶GT EQPVCKPKPI VJG CEVWCN GNGOGPVU QH VJG CTTC[0QTOCN
YKNN CEEGUU VJG GNGOGPVU KP CP CTTC[WUKPI KPFGZKPI HCEKNKVKG

"O FYBNQMF

```
   JNQPSUVNQBZTOQ
   B  OQBSSBOHF SFTIBQF
   B
BSSBZ <<              >
     <               >
     <              >>
   B TIBQF

   B OEJN

   B EUZQØFBNF
JOU
   B JUFNTJ[F

   B TJ[F

   UZQB
DMBTT OVNQZ OEBSSBZ
   C  OQBSSBZ <   >
   C
BSSBZ <      >
   UZQC
DMBTT OVNQZ OEBSSBZ
```

"SSBZ $SFBUJPO

6JGTG CTG UGXGTCN YC[U VQ ETGCVG CTTC[U

(QT GZCORNG [QW ECP ETGCVG CP CTTC[HTQO B SSBZWPEVKQP JQGNK[UV QT
TGUWNVKPI CTTC[KU FGFWEGF HTQO VJG V[RG QH VJG GNGOGPVU KP VJG UG

```
JNQPSU VNQZBTOQ
B   OQBSSBZ <    >
B
BSSBZ <       >
B EUZQF
EUZQF  JOU
C   OQBSSBZ <           >
C EUZQF
EUZQF  GMPBU
```

HTGSWGPV GTTQT BSSBZ KPXQNXKPI ECNNKPI CTTC[YKVJ OWNVKRNG PWOGTKE CTIWOGPVU TCVJGT VJCP RTQXKF
OGPV

```
B   OQBSSBZ                 830/(
5SBDFCBDL  NPTU SFDFOU DBMM MBTU

5ZQF&SSPS BSSBZ   UBLFT GSPN   UP   QPTJUJPOBM BSHVNFOUT CVU   XFSF HJWFO
B   OQBSSBZ <       >     3*()5
```

BSSBZ VTCPUHQTOU UGSWGPEGU QH UGSWGPEGU KPVQ VYQ FKOGPUKQPCN C
FKOGPUKQPCN CTTC[U CPF UQ QP

```
C   OQBSSBZ <                  >
C
BSSBZ <<             >
      <            >>
```

6JG V[RG QH VJG CTTC[ECP CNUQ DG GZRNKEKVN[URGEKÆGF CV ETGCVKQP V

```
D   OQBSSBZ <<>  <    >>  EUZQFOQDPNQMFY
D
BSSBZ <<     K       K>
      <     K       K>>
```

1HVGP VJG GNGOGPVU QH CP CTTC[CTG QTKIKPCNN[WPMPQYP DWV KVU UK
ETGCVG CTTC[U YKVJ KPKVKCN RNCEGJQNFGT EQPVGPVU 6JGUG OKPKOK\G V

6JG HWPEVKQP \GTQU ETGCVGU CP CTTC[HWNN QH \GTQU VJG HWPEVKQP QPGU
FNQUETGCVGU CP CTTC[YJQUG KPKVKCN EQPVGPV KU TCPFQO CPF FGRGPFU
QH VJG ETGCVGF CTTC[KU HNQCV ECP DG URGEKÆGF XKC VJG MG[YQTF CTIWOGPV

```
OQ[FSPT
BSSBZ <<             >
      <            >
      <            >>
OQPOFT          EUZQFOQJOU
BSSBZ <<<         >
       <         >
       <         >>

      <<         >
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

```
      <          >
      <          >>>  EUZQF JOU
  OQFNQUZ
BSSBZ <<        F          F          F   >     NBZ WBSZ
     <        F          F          F  >>
```

6Q ETGCVG UGSWGPEGU QH PWODGTU PWORZ RTQXKFGU C HWPEVKQP CPCNQIQWU VQ VJG 2[VJQP DWKNV-KP TCPIG
DWV TGVWTPU CP CTTC[

```
  OQBSBOHF
BSSBZ <             >
  OQBSBOHF          JU BDDFQUT GMPBU BSHVNFOUT
BSSBZ <                       >
```

9JGP BSBOHF KU WUGF YKVJ ¸QCVKPI RQKPV CTIWOGPVU KV KU IGPGTCNN[PQV RQUUKDNG VQ RTGFKEV VJG PWODGT QH GNGOGPVU QDVCKPGF FWG VQ VJG ·PKVG ¸QCVKPI RQKPV RTGEKUKQP MJOTQBDF VJKU TGCUQP KV KU WUWCNN[DGVVGT VQ WUG VJG HWPEVKQP TGEGKXGU CU CP CTIWOGPV VJG PWODGT QH GNGOGPVU VJCV YG YCPV KPUVGCF QH VJG UVGR

```
  GSPN OVNQZ JNQPSU QU
  OQMJOTQBDF                    OVNCFST GSPN  UP
BSSBZ <                         >
  Y  OQMJOTQBDF   QJ            VTFGVM UP FWBMVBUF GVODUJPO BU MPUT
 QPJOUT
  G  OQTJO Y
```

4FF BMTP

BSSBZ [FSPT [FSPT@MJLF POFT POFT@MJLF FNQUZ FNQUZ@MJLF BSBOHF MJOTQBDF
OVNQZ SBOEPN (FOFSBUPS SBOE OVNQZ SBOEPN (FOFSBUPS SBOEO GSPNGVODUJPO GSPNGJMF

1SJOUJOH "SSBZT

9JGP [QW RTKPV CP CTTC[0WO2[FKURNC[U KV KP C UKOKNCT YC[VQ PGUVGF NKUVU DWV YKVJ VJG HQNNQYKPI NC[QWV

YVJG NCUV CZKU KU RTKPVGF HTQO NGHV VQ TKIJV

YVJG UGEQPF VQ NCUV KU RTKPVGF HTQO VQR VQ DQVVQO

YVJG TGUV CTG CNUQ RTKPVGF HTQO VQR VQ DQVVQO YKVJ GCEJ UNKEG UGRCTCVGF HTQO VJG PGZV D[CP GORV[NKPG

1PG FKOGPUKQPCN CTTC[U CTG VJGP RTKPVGF CU TQYU DKFKOGPUKQPCNU CU OCVTKEGU CPF VTKFKOGPUKQPCNU CU NKUVU QH OCVTKEGU

```
  B  OQBSBOHF               E BSSBZ
  QSJOU B
<          >

  C  OQBSBOHF  SFTIBQF      E BSSBZ
  QSJOU C
<<        >
 <        >
 <        >
 <        >>

  D  OQBSBOHF  SFTIBQF      E BSSBZ
  QSJOU D
<<<          >
```

EQPVKPWGU QP PGZV RCIG

EQPVKPWGF HTQO RTGXKQWU

```
<   >
<   >>

<<   >
<   >
<   >>>
```

5GG GF MPQX IGV OQTGSFFGTIECKFNU QP

+H CP CTTC[KU VQQ NCTIG VQ DG RTKPVGF 0WO2[CWVQOCVKECNN[UMKRU V

```
QSJOU OQBSBOHF
<   >

QSJOU OQBSBOHF   SFTIBQF
<<   >
<   >
<   >

<   >
<   >
<   >>
```

6Q FKUCDNG VJKU DGJCXKQWT CPF HQTEG 0WO2[VQ RTKPV VJG GPVKTG
TFU@QSJOUPQUJPOT

```
OQTFU@QSJOUPQUJPOT UISFTIPME  TZT NPEVMF TIPVME CF JNQPSUFE
```

#BTJD 0QFSBUJPOT

#TKVJOGVKE QRGTCVQTU QP CTTC[U CRRN[ETGCVGF CPF ·NNGF YKVJ VJG T

```
B  OQBSSBZ <   >
C  OQBSBOHF
C
BSSBZ <   >
D  B  C
D
BSSBZ <   >
C
BSSBZ <   >
OQTJO B
BSSBZ <   >
B
BSSBZ < 5SVF  5SVF  'BMTF  'BMTF>
```

7PNKMG KP OCP[OCVTKZ NCPIWCIGU VJG RTQFWEV QRGTCVQT KU 0WO2[CTTC[
ECP DG RGTHQTOGF WUKPI VJG FQV HWPEVKQP QT OGVJQF

```
"  OQBSSBZ <<>
        <  >>
#  OQBSSBZ <<>
        <  >>
"  #    FMFNFOUXJTF QSPEVDU
BSSBZ <<   >
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

```
        <     >>
    " ! #        NBUSJY QSPEVDU
BSSBZ <<     >
        <     >>
    " EPU #    BOPUIFS NBUSJY QSPEVDU
BSSBZ <<     >
        <     >>
```

5QOG QRGTCVKQPU UWEJ CU CPF CEV KP RNCEG VQ OQFKH[CP GZKUVKPI CTTC[TCVJGT V

```
    SH   OQSBOEPN FNFGBVMU@SOH  DSFBUF JOTUBODF PG EFGBVMU SBOEPN OVNCFS
 HFOFSBUPS
    B   OQPOFT       EUZQF JOU
    C   SHSBOEPN
    B
    B
BSSBZ <<         >
        <         >>
    C     B
    C
BSSBZ <<                                         >
        <                                    >>
    B     C    C JT OPU BVUPNBUJDBMMZ DPOWFSUFE UP JOUFHFS UZQF
5SBDFCBDL   NPTU SFDFOU DBMM MBTU

OVNQZ DPSF @FYDFQUJPOT @6'VOD0VUQVU$BTUJOH&SSPS $BOOPU DBTU VGVOD  BEE  PVUQVU GSPN
 EUZQF  GMPBU       UP EUZQF  JOU       XJUI DBTUJOH SVMF  TBNF@LJOE
```

9JGP QRGTCVKPI YKVJ CTTC[U QH FKHGTGPV V[RGU VJG V[RG QH VJG TGUWNVKPI CTTC[EQTTGURQPFU VQ VJG OQTG IGPGTCN QT RTGEKUG QPG C DGJCXKQT MPQYP CU WRECUVKPI

```
    B   OQPOFT  EUZQF JOU
    C   OQMJOTQBDF QJ
    C EUZQF OBNF
 GMPBU
    D    B    C
    D
BSSBZ <                                >
    D EUZQF OBNF
 GMPBU
    E   OQFYQ  D   K
    E
BSSBZ <                      K                         K
                           K>
    E EUZQF OBNF
 DPNQMFY
```

/CP[WPCT[QRGTCVKQPU UWEJ CU EQORWVKPI VJG UWO QH CNN VJG GNGOGPVU KP VJG CTTC[CTG KORNGOGPVGF CU OGVJQFU QH VJG OEBSSBZ ENCUU

```
    B   SHSBOEPN
    B
BSSBZ <<                                         >
        <                                    >>
    B TVN
```

EQPVKPWGU QP PGZV RCIG

EQPVKPWGF HTQO RTGXKQWU

```
BNJO

BNBY
```

$[FGHCWNV VJGUG QRGTCVKQPU CRRN[VQ VJG CTTC[CU VJQWIJ KV YGTG C
URGEKHKPI VJG CZKU RCTCOGVGT [QW ECP CRRN[CP QRGTCVKQP CNQPI VJG URGEK·G

```
C OQBSBOHF SFTIBQF
C
BSSBZ <<            >
    <            >
    <            >>

C TVN BYJT      TVN PG FBDI DPMVNO
BSSBZ <            >

CNJO BYJT       NJO PG FBDI SPX
BSSBZ <      >

C DVNTVN BYJT   DVNVMBUJWF TVN BMPOH FBDI SPX
BSSBZ <<            >
    <            >
    <            >>
```

6OJWFSTBM 'VODUJPOT

0WO2[RTQXKFGU HCOKNKCT OCVJGOCVKECN HWPEVKQPU UWEJ CU UKP EQ
VKQP VKQP 9KVJKP 0WO2[VJGUG HWPEVKQPU QRGTCVG GNGOGPVYKUG QP C

```
# OQBSBOHF
#
BSSBZ <      >
OQFYQ #
BSSBZ <                       >
OQTRSU #
BSSBZ <                        >
$ OQBSSBZ <        >
OQBEE # $
BSSBZ <         >
```

4FF BMTP

BMM BOZ BQQMZ@BMPOH BYJT BSHNBY BSHTPSU BWFSBHF CJODPVOU DPSSDPFG DPW DSPTT DVNQSPE DVNTVN EJGG EPU GMPPS JOOFS JOW MFYTPSU NBY NBYJNVN NFBO NFEJBO NJO NJOJNVN OPO[FSP PVUFS QSPE SF SPVOE TPSU TUE TVN USBDF USBOTQPTF WBS WEPU WFDUPSJ[F XIFSF

*OEFYJOH 4MJDJOH BOE *UFSBUJOH

0OF EJNFOTJPOBM BSSBZT DBO CF JOEFYFE TMJDFE BOE JUFSBUFE PWFS NVDI MJLF MJTUT BOE PUIFS 1ZUIPO TFRVFODFT

```
B OQBSBOHF
B
BSSBZ <                                              >
B<>

B<  >
BSSBZ <        >
   FRVJWBMFOU UP B<    >
   GSPN TUBSU UP QPTJUJPO    FYDMVTJWF  TFU FWFSZ  OE FMFNFOU UP
B<  >
B
BSSBZ <                                                      >
B<  >   SFWFSTFE B
BSSBZ <                                                      >
GPS J JO B
   QSJOU J
```

.VMUJEJNFOTJPOBM BSSBZT DBO IBWF POF JOEFY QFS BYJT 5IFTF JOEJDFT BSF HJWFO JO B UVQMF TFQBSBUFE CZ DPNNBT

```
EFG G Y  Z
   SFUVSO   Y   Z

C  OQGSPNGVODUJPO G  EUZQFJOU
C
BSSBZ <<          >
   <          >
   <          >
   <          >
   <          >>
C<  >

C<    >   FBDI SPX JO UIF TFDPOE DPMVNO PG C
BSSBZ <             >
C<  >     FRVJWBMFOU UP UIF QSFWJPVT FYBNQMF
BSSBZ <             >
C<    >   FBDI DPMVNO JO UIF TFDPOE BOE UIJSE SPX PG C
BSSBZ <<          >
   <          >>
```

8IFO GFXFS JOEJDFT BSF QSPWJEFE UIBO UIF OVNCFS PG BYFT UIF NJTTJOH JOEJDFT BSF DPOTJEFSFE DPNQMFUF TMJDFT

```
C<>     UIF MBTU SPX  &RVJWBMFOU UP C<    >
BSSBZ <              >
```

6JG GZRTGUUKQP YK<VJKU TGVWTPU OCP[CU OCP[GNGOGPVU CU VJG CTTC[...
TGOCKPKPI CZGU 0WO2[CNUQ CNNQYU [QW VQ YTKVG VJKU WUKPI FQVU CU

6JG FQVU TGRTGUGPV CU OCP[EQNQPU CU PGGFGF VQ RTQFWEG C EQORNGVG
YKVJ CZGU VJGP

YY< >KU GSWKX<CNGPV VQ >

YY< >VQ< >CPF

YY< >VQ< >

```
D OQBSSBZ <<<   >   B % BSSBZ UXP TUBDLFE % BSSBZT
        <      >>
       <<       >
        <       >>>
D TIBQF

D<   >  TBNF BT D<    > PS D< >
BSSBZ <<        >
    <       >>
D<   >  TBNF BT D<    >
BSSBZ <<     >
    <     >>
```

*UFSBUJOH overall multidimensional arrays is done with respect to the ·T...

```
GPS SPX JO C
    QSJOU SPX

<     >
<        >
<        >
<        >
<        >
```

*QYGXGT KH QPG YCPVU VQ RGTHQTO CP QRGTCVKQP QP GCEJ GNGOGPV KP VJG CTTC[...
KVGTCVQT QXGT CNN VJG GNGOGPVU QH VJG CTTC[

```
GPS FMFNFOU JO C GMBU
    QSJOU FMFNFOU
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

4FF BMTP

*OEFYJOH PO OEBSSBZT CTTC[U KPFGZKPIX BTOHGTOCPF GSJOEDFT

4IBQF .BOJQVMBUJPO

$IBOHJOH UIF TIBQF PG BO BSSBZ

#P CTTC[JCU C UJCRG IKXGP D[VJG PWODGT QH GNGOGPVU CNQPI GCEJ CZK

```
B OQGMPPS SHSBOEPN
B
BSSBZ <<           >
    <           >
    <           >>
B TIBQF
```

6JG UJCRG QH CP CTTC[ECP DG EJCPIGF YKVJ XCTKQWU EQOOCPFU 0QVG
OQFK·GF CTTC[DWV FQ PQV EJCPIG VJG QTKIKPCN CTTC[

```
B SBWFM  SFUVSOT UIF BSSBZ GMBUUFOFE
BSSBZ <                             >
B SFTIBQF     SFUVSOT UIF BSSBZ XJUI B NPEJGJFE TIBQF
BSSBZ <<    >
    <    >
    <    >
    <    >
    <    >
    <    >>
B 5   SFUVSOT UIF BSSBZ USBOTQPTFE
BSSBZ <<       >
    <       >
    <       >
    <       >>
B 5 TIBQF

B TIBQF
```

6JG QTFGT QH VJG GNGOGPVU SBWFM CTTC[KU %UV[NG VJCV KU VJG TK
VJG HCUVGUV UQ VJG GNGOGPV CHVGT C>+ KU C>+ CTTC[KU TGUJCRGF VQ UQOG QVJG
KU VTGCVGF CU S% UV[NG 0WO2[PQTOCNN[ETGCVGU CTTC[U UVQTGF KP
CTIWOGPV DWV KH VJG CTTC[YCU OCFG D[VCMKPI UNKEGU QH CPQVJGT CT
EQRKGF 6JG SBWFM CPF SFTIBQF CP CNUQ DG KPUVTWEVGF WUKPI CP QRVKQPCN C
CTTC[U KP YJKEJ VJG NGHVOQUV KPFGZ EJCPIGU VJG HCUVGUV

6JG SFTIBQF HWPEVKQP TGVWTPU KVU CTIWOGPV YKVJ C OQFK·GF UJCRG YJGTGC
VJG CTTC[KVUGNH

```
    B
BSSBZ <<          >
      <          >
      <          >>
    B SFTJ[F
    B
BSSBZ <<                 >
      <                 >>
```

+H C FKOGPUKQP KU IKXGP CU KP C TGUJCRG QRGTCVKQP VJG QVJGT FKOGPUKQPU CTG

```
    B SFTIBQF
BSSBZ <<          >
      <          >
      <          >>
```

4FF BMTP

OEBSSBZ TIBQF OEBSSBZ SFTIBQF OEBSSBZ SFTJ[F SFTJ[F BWFM

4UBDLJOH UPHFUIFS EJGGFSFOU BSSBZT

5GXGTCN CTTC[U ECP DG UVCEMGF VQIGVJGT CNQPI FK¶GTGPV CZGU

```
    B  OQGMPPS  SHSBOEPN
    B
BSSBZ <<     >
      <     >>
    C  OQGMPPS  SHSBOEPN
    C
BSSBZ <<     >
      <     >>
    OQWTUBDL  B  C
BSSBZ <<     >
      <     >
      <     >
      <     >>
    OQITUBDL  B  C
BSSBZ <<          >
      <          >>
```

6JG HWPEVKQP EQNWOP@UVCEM UVCEMU & CTTC[U CU EQNWOPU KPVQ C & CTTC[+V KU G CTTC[U

```
    GSPN OVNQZ JNQPSU OFXBYJT
    OQDPMVNO@TUBDL  B XJUI  % BSSBZT
BSSBZ <<          >
      <          >>
    B  OQBSSBZ <  >
    C  OQBSSBZ <  >
    OQDPMVNO@TUBDL  B SFUVSOT B  % BSSBZ
BSSBZ <<     >
      <     >>
    OQITUBDL  B  C        UIF SFTVMU JT EJGGFSFOU
BSSBZ <         >
    B<  OFXBYJT>WJFX ABA BT B  % DPMVNO WFDUPS
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

```
BSSBZ << >
< >>
OQDPMVNO@TUBDL B< OFXBYJT> C< OFXBYJT>
BSSBZ << >
< >>
OQITUBDL B< OFXBYJT> C< OFXBYJT> UIJT SFTVMU JT UIF TBNF
BSSBZ << >
< >>
```

1P VJG QVJGT JCPF SPX@TUBDL UVBDMU VJG KPRWV CTTC[U PQ [KPRWV CTTC[U]
CNKOWUHUGB TDL

```
OQDPMVNO@TUBDL JT OQITUBDL
'BMTF
OQSPX@TUBDL JT OQWTUBDL
5SVF
```

+P IGPGTCN HQT CTTC[U YKVJ OQTG VJCP VYQ FKOGPUKQPU JUVBDL UVCEMU CNQPI VJGKT ·TUV CZKU EQPECVGPCVG CNNQYU HQT CP QRVKQPCN CTIWOGPV IKXKPI VJG PWODGT QH VJG CZKU CNQPI YJKEJ VJG EQPECVGPCVKQP UJQWNF JCRRGP

/PUF

+P EQORNGZ ECUGU SOE EQPEGTU WUGHWN HQT ETGCVKPI CTTC[U D[UVCEMKPI PWODGTU CNQPI QPG CZKU
TCPIG NKVGTCNU

```
OQS@< >
BSSBZ < >
```

9JGP WUGF YKVJ CTTC[U CU CTIWOGPVU S@ CPF D@ UVCEM CTTC[U CNQPI VJGKT ·TUV CZKU
HQT CP QRVKQPCN CTIWOGPVU IKXKPI VJG PWODGT QH VJG CZKU CNQPI YJKEJ

4FF BMTP

ITUBDL WTUBDL DPMVNO@TUBDL DPODBUFOBUF D@ S@

4QMJUUJOH POF BSSBZ JOUP TFWFSBM TNBMMFS POFT

7UKPI TQMJU QW ECP URNKV CP CTTC[CNQPI KVU JQTK\QPVCN CZKU GKVJGT D[URGEKH[KPI VJG PWODGT QH GSWCNN[UJCRGF CTTC[U VQ TGVWTP QT D[URGEKH[KPI VJG EQNWOPU CHVGT YJKEJ VJG FKXKUKQP UJQWNF QEEWT

```
B OQGMPPS SHSBOEPN
B
BSSBZ << >
< >>
4QMJU ABA JOUP
OQITQMJU B
<BSSBZ << >
< >> BSSBZ << >
< >> BSSBZ << >
< >> >
4QMJU ABA BGUFS UIF UIJSE BOE UIF GPVSUI DPMVNO
OQITQMJU B
<BSSBZ << >
< >> BSSBZ << >
```

EQPVKPWGU QP PGZV RCIG

EQPVKPWGF HTQO RTGXKQWU

```
    < >>   BSSBZ <<                          >
    <                          >> >
```

WTQMURNKVU CNQPI VJG BSSBZ@ETQNNQKU QPG VQ URGEKH[CNQPI YJKEJ CZK

$PQJFT BOE 7JFXT

9JGP QRGTCVKPI CPF OCPKRWNCVKPI CTTC[U VJGKT FCVC KU UQOGVKOGU QHVGP C UQWTEG QH EQPHWUKQP HQT DGIKPPGTU 6JGTG CTG VJTGG ECUGU

/P $PQZ BU "MM

5KORNG CUUKIPOGPVU OCMG PQ EQR[QH QDLGEVU QT VJGKT FCVC

```
  B    OQBSSBZ <<          >
               <          >
               <          >>
  C    B              OP OFX PCKFDU JT DSFBUFE
  C JT B              B BOE C BSF UXP OBNFT GPS UIF TBNF OEBSSBZ PCKFDU
5SVF
```

2[VJQP RCUUGU OWVCDNG QDLGEVU CU TGHGTGPEGU UQ HWPEVKQP ECNNU

```
  EFG G Y
    QSJOU JE Y

  JE B      JE JT B VOJRVF JEFOUJGJFS PG BO PCKFDU
        NBZ WBSZ
  G B
        NBZ WBSZ
```

7JFX PS 4IBMMPX $PQZ

&K¶GTGPV CTTC[QDLGEVU ECP UJCTG VJG UCOG FCVC 6JG XKGY OGVJQF ETGCVGU C PGY CTTC[QDLGEV VJCV

```
  D    B WJFX
  D JT B
'BMTF
  D CBTF JT B              D JT B WJFX PG UIF EBUB PXOFE CZ B
5SVF
  D GMBHT PXOEBUB
'BMTF

  D    D SFTIBQF            B T TIBQF EPFTO U DIBOHF
  B TIBQF

  D<    >                  B T EBUB DIBOHFT
  B
BSSBZ <<                   >
      <                 >
      <                 >>
```

5NKEKPI CP CTTC[TGVWTPU C XKGY QH KV

```
    T    B<       >
    T< >          T< > JT B WJFX PG T  /PUF UIF EJGGFSFODF CFUXFFO T        BOE T
    B
BSSBZ <<                        >
       <                      >
       <                      >>
```

%FFQ $PQZ

6JG OPQQZ GVJQF OCMGU C EQORNGVG EQR[QH VJG CTTC[CPF KVU FCVC

```
    E    B DPQZ     B OFX BSSBZ PCKFDU XJUI OFX EBUB JT DSFBUFE
    E JT B
'BMTF
    E CBTF JT B    E EPFTO U TIBSF BOZUIJOH XJUI B
'BMTF
    E<    >
    B
BSSBZ <<                        >
       <                      >
       <                      >>
```

5QOGVKOGU EQR[UJQWNF DG ECNNGF CHVGT UNKEKPI KH VJG QTKIKPCN CTTC[KU PQ C JWIG KPVGTOGFKCVG TGUWNV CPF C OGOQT[EQPUWORVKQP CPF OG KU PQ EQPUVTWEVKQP UNKEKPI

```
    B    OQ BSSBOHF JOUF
    C    B<    > DPQZ
    EFM B    UIF NFNPSZ PG AABAA DBO CF SFMFBTFE
```

+H   B<   KU WUGF KPUVGCF UQOG OGOQT[KU PGXGT DG HTGGF

'VODUJPOT BOE .FUIPET 0WFSWJFX

*GTG KU C NKUV QH UQOG WUGHWN 0WO2[HWPEVKQPU CPF OGVJQFU PCOGU

"SSBZ $SFBUJPO

BSBOHF BSSBZ DPQZ FNQUZ FNQUZ@MJLF FZF GSPNGJMF GSPNGVODUJPO JEFOUJUZ MJOTQBDF MPHTQBDF NHSJE PHSJE POFT POFT@MJLF S@ [FSPT [FSPT@MJLF

$POWFSTJPOT

OEBSSBZ BTUZQF BUMFBTU@E BUMFBTU@E BUMFBTU@E NBU

.BOJQVMBUJPOT

BSSBZ@TQMJU DPMVNO@TUBDL DPODBUFOBUF EJBHPOBM ETQMJU ETUBDL IQMJU ITUBDL OEBSSBZ JUFN OFXBYJT SBWFM SFQFBU SFTIBQF SFTJ[F TRVFF[F TXBQBYFT UBLF USBOTQPTF WTQMJU WTUBDL

2VFTUJPOT

BMM BOZ OPO[FSP XIFSF

0SEFSJOH

BSHNBY BSHNJO BSHTPSU NBY NJO QUQ TFBSDITPSUFE

0QFSBUJPOT

DIPPTF DPNQSFTT DVNQSPE DVNTVN JOOFS OEBSSBZ GJMM JNBH QSPE QVU QVUNBTL SFBM
TVN

#BTJD 4UBUJTUJDT

DPW NFBO TUE WBS

#BTJD -JOFBS "MHFCSB

DSPTT EPU PVUFS MJOBMH TWE WEPU

-FTT #BTJD

#SPBEDBTUJOH SVMFT

$TQCFECUVKPI CNNQYU WPKXGTUCN HWPEVKQPU VQ FGCN KP C OGCPKPIHW

6JG ·TUV TWNG QH DTQCFECUVKPI KU VJCV KH CNN KPRWV CTTC[U FQ PQV JCX
RTGRGPFGF VQ VJG UJCRGU QH VJG UOCNNGT CTTC[U WPVKN CNN VJG CTTC[

6JG UGEQPF TWNG QH DTQCFECUVKPI GPUWTGU VJCV CTTC[U YKVJ C UK\G Q
UK\G QH VJG CTTC[YKVJ VJG NCTIGUV UJCRG CNQPI VJCV FKOGPUKQP 6JG X
CNQPI VJCV FKOGPUKQP HQT VJG SDTQCFECUVT CTTC[

#HVGT CRRNKECVKQP QH VJG DTQCFECUVKPI TWNGU VJG UK\GU #SPBEDBTUJOH

"EWBODFE JOEFYJOH BOE JOEFY USJDLT

0WO2[Q¶GTU OQTG KPFGZKPI HCEKNKVKGU VJCP TGIWNCT 2[VJQP UGSWGPI
YG UCY DGHQTG CTTC[U ECP DG KPFGZGF D[CTTC[U QH KPVGIGTU CPF CTTC

*OEFYJOH XJUI "SSBZT PG *OEJDFT

```
    B   OQBSBOHF        UIF GJSTU    TRVBSF OVNCFST
    J   OQBSSBZ <          >     BO BSSBZ PG JOEJDFT
    B<J>  UIF FMFNFOUT PG ABA BU UIF QPTJUJPOT AJA
BSSBZ <                   >

    K   OQBSSBZ <<> <   >>     B CJEJNFOTJPOBM BSSBZ PG JOEJDFT
    B<K>  UIF TBNF TIBQF BT AKA
BSSBZ <<      >
       <      >>
```

9JGP VJG KPFGZGF CTTC[B KU OWNVKFKOGPUKQPCN C UKPING CTTC[QH KPFKEGU TGHGTU VQ VJG HKTUV FKOGPUKQP
GZCORNG UJQYU VJKU DGJCXKQT D[EQPXGTVKPI CP KOCIG QH NCDGNU KPVQ

```
    QBMFUUF QBSSBZ <<    >            CMBDL
                  <         >         SFE
                  <         >         HSFFO
                  <         >         CMVF
                  <             >>    XIJUF
    JNBHF OQBSSBZ <<       >     FBDI WBMVF DPSSFTQPOET UP BÃDPMPS JO UIF
QBMFUUF
                <           >>
    QBMFUUF<JNBHF>              DPMPS JNBHF
BSSBZ <<<             >
      <            >
      <            >
      <            >>

     <<             >
      <            >
      <            >
      <            >>>
```

9G ECP CNUQ IKXG KPFGZGU HQT OQTG VJCP QPG FKOGPUKQP 6JG CTTC[U C
UJCRG

```
    B    OQBSBOHF   SFTIBQF
    B
BSSBZ <<              >
     <             >
     <             >>
    J    OQBSSBZ << >     JOEJDFT GPS UIF GJSTU EJN PG ABA
               <    >>
    K    OQBSSBZ << >     JOEJDFT GPS UIF TFDPOE EJN
               <    >>

    B<J   K>  J BOE K NVTU IBWF FRVBM TIBQF
BSSBZ <<      >
     <      >>

    B<J  >
BSSBZ <<      >
     <      >>

    B<    K>
BSSBZ <<<      >
      <      >>

     <<      >
      <      >>

     <<      >
      <      >>>
```

+P 2[VJQP B<J K> GZCEVN[VJG UCOG CU ... YG EQP ... VJG ... FQ VJG
KPFGZKPI YKVJ VJCV

```
    M    J   K
      FRVJWBMFOU UP B<J   K>
    B<M>
BSSBZ <<      >
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

```
      <      >>
```

*QYGXGT YG ECP PQV JFQPWKXPVQ[CPWCTTCP[IDGECWUG VJKU CTTC[YKNN DG K
FKOGPUBKQP QH

```
    T   OQBSSBZ <J  K>
      OPU XIBU XF XBOU
    B<T>
5SBDFCBDL  NPTU SFDFOU DBMM MBTU
  'JMF TUEJO MJOF JO NPEVMF
*OEFY&SSPSEFY  JT PVU PG CPVOETGPS BYJT  XJUI TJ[F
      TBNF BT AB<J  K>A
    B<UVQMF>
BSSBZ <<      >
      <      >>
```

#PQVJGT EQOOQP WUG QH KPFGZKPI YKVJ CTTC[U KU VJG UGCTEJ QH VJG OC

```
    UJNF OQMJOTQBDF          UJNF TDBMF
    EBUB OQTJO OBQSBOHF  SFTIBQF        UJNF EFQFOEFOU TFSJFT
    UJNF
BSSBZ <                                 >
    EBUB
BSSBZ <<                                    >
      <                                     >
      <                                     >
      <                                     >
      <                                     >>
      JOEFY PG UIF NBYJNB GPS FBDI TFSJFT
    JOE EBUBSHNBY BYJT
    JOE
BSSBZ <         >
      UJNFT DPSSFTQPOEJOH UP UIF NBYJNB
    UJNF@NBUJNF<JOE>

    EBUB@NBEYBUB<JOSEBOHEBUBTIBQF< >      EBUB<JOE< >   > EBUB<JOE< >   >
,
    UJNF@NBY
BSSBZ <                         >
    EBUB@NBY
BSSBZ <                                 >
    OQBMM EBUB@NBEBYUBNBY BYJT
5SVF
```

;QW ECP CNUQ WUG KPFGZKPI YKVJ CTTC[U CU C VCTIGV VQ CUUKIP VQ

```
    B   OQBSBOHF
    B
BSSBZ <           >
    B<<      >>
    B
BSSBZ <           >
```

*QYGXGT YJGP VJG NKUV QH KPFKEGU EQPVCKPU TGRGVKVKQPU VJG CUUK

```
B OQBSBOHF
B<< >> < >
B
BSSBZ < >
```

6JKU KU TGCUQPCDNG GPQWIJ DWV YCVEJ QWV HQT UWDVNG RQKPVU [illegible]

```
B OQBSBOHF
B<< >>
B
BSSBZ < >
```

'XGP VJQWIJ QEEWTU VYKEG KP VJG NKUV QH KPFKEGU VJG VJ GNGOGPV TGSWKTGU VQ DG GSWKXCNGPV VQ

*OEFYJOH XJUI #PPMFBO "SSBZT

9JGP YG KPFGZ CTTC[U YKVJ CTTC[U QH KPVGIGT KPFKEGU YG CTG RTQXKF VJG CRRTQCEJ KU FK¶GTGPV YG GZRNKEKVN[EJQQUG YJKEJ KVGOU KP VJG

6JG OQUV PCVWTCN YC[QPG ECP VJKPM QH HQT DQQNGCP KPFKEGU [illegible] VQ QTKIKPCN CTTC[

```
B OQBSBOHF SFTIBQF
C B
C ACA JT B CPPMFBO XJUI ABA T TIBQF
BSSBZ <<'BMTF 'BMTF 'BMTF 'BMTF>
<'BMTF 5SVF 5SVF 5SVF>
< 5SVF 5SVF 5SVF 5SVF>>
B<C> E BSSBZ XJUI UIF TFMFDUFE FMFNFOUT
BSSBZ < >
```

6JKU RTQRGTV[ECP DG XGT[WUGHWN KP CUUKIPOGPVU

```
B<C> "MM FMFNFOUT PG ABA IJHIFS UIBO CFDPNF
B
BSSBZ << >
< >
< >>
```

;QW ECP NQQM CV VJG HQNNQYKPI GZCORNG VQ UGG JQY VQ WUG DQQNGCP [illegible]

```
JNQPSU NBUQMPUMJC QZQMPU BT QMU
JNQPSU OVNQZ BT OQ
EFG NBOEFMCSPU NBYJU S
    3FUVSOT BO JNBHF PG UIF .BOEFMCSPU GSBDUBM PG TJ[F I X
    Y OQMJOTQBDF I
    Z OQMJOTQBDF X
    " # OQNFTIHSJE Y Z
    $ " # K
    [ OQ[FSPT@MJLF $
    EJWUJNF NBYJU OQ[FSPT [TIBQF EUZQF JOU

    GPS J JO SBOHF NBYJU
        [ [ $
        EJWFSHF [ S XIP JT EJWFSHJOH
```

EQPVKPWGU QP PGZV RCI [illegible]

EQPVKPWGF HTQO RTGXKQWU

```
        EJW@OPX EJWFSHFEJWUJNFNBYJU    XIP JT EJWFSHJOH OPX
        EJWUJNF<EJW@JOPX>              OPUF XIFO
        [<EJWFSHFS>                    BWPJE EJWFSHJOH UPP NVDI

    SFUVSO EJWUJNF
QMU JDMG
QMU JNTIPX NBOEFMCSPU
```

6JG UGEQPF YC[QH KPFGZKPI YKVJ DQQNGCPU KU OQTG UKOKNCT VQ KPVGIG
& DQQNGCP CTTC[UGNGEVKPI VJG UNKEGU YG YCPV

```
B   OQBSBOHF  SFTIBQF
C    OQBSSBZ<BMTF5SVF5SVF>          GJSTU EJN TFMFDUJPO
C    OQBSSBZ<5SVF'BMTF5SVF'BMTF>    TFDPOE EJN TFMFDUJPO

B<C     >                           TFMFDUJOH SPXT
BSSBZ <<              >
      <              >>

B<C >                               TBNF UIJOH
BSSBZ <<              >
      <              >>

B<    C >                           TFMFDUJOH DPMVNOT
BSSBZ <<      >
      <      >
      <      >>

B<C    C >                          B XFJSE UIJOH UP EP
BSSBZ <      >
```

0QVG VJCV VJG NGPIVJ QH VJG & DQQNGCP CTTC[OWUV EQKPEKFG YKVJ VJG
+P VJG RTGXKQWU GZCORNG VJG KPFKEGU QH NGPIVJ KU UWKVCDNG VQ KP
EQNWOB PU QH

5IF JY@ GVODUJPO

6JG JY@ WPEVKQP ECP DG WUGF VQ EQODKPG FK¶GTGPV XGEVQTU UQ CU VQ Q
[QW YCPV VQ EQORWVG CNN VJG C D E HQT CNN VJG VTKRNGVU VCMGP HTQO

```
    B  OQBSSBZ <      >
    C  OQBSSBZ <    >
    D  OQBSSBZ <        >
    BY  CY  DYOQJY@ B  C  D
    BY
BSSBZ <<< >>

      << >>

      << >>

      << >>>
    CY
BSSBZ <<< >
       < >
       < >>>
    DY
BSSBZ <<<          >>>
    BYTIBQF  CTYIBQF  DTYIBQF

    SFTVMUBY  CY  DY
    SFTVMU
BSSBZ <<<              >
       <              >
       <              >>

      <<              >
       <              >
       <              >>

      <<              >
       <              >
       <              >>

      <<              >
       <              >
       <              >>>
    SFTVMU<   >

    B<>  C<>  D<>
```

;QW EQWNF CNUQ KORNGOGPV VJG TGFWEG CU HQNNQYU

```
    EFGVGVOD@SFVEGVDUF WFDUPST
      WT  OQJY@WFDUPST
      S  VGDUJEFOUJUZ
      GPSWJOWT
        S  VGDU S  W
      SFUVSSO
```

CPF VJGP WUG KV CU

```
   VGVOD@SFEVBDEFOQ  C  D
BSSBZ <<<              >
       <               >
       <               >>

      <<               >
       <               >
       <               >>

      <<               >
       <               >
       <               >>

      <<               >
       <               >
       <               >>>
```

6JG CFXCPVCIG QH VJKU XGTUKQP QH TGFWEG EQORCTGF VQ VJG QTKIKPCN SVKPTQTFGT VQ CXQKF ETGCVKPI CP CTIWOGPV CTTC[VJG UK\G QH VJG QWV

*OEFYJOH XJUI TUSJOHT

5GGUSVDUVSFE BSSBZT

5SJDLT BOE 5JQT

*GTG YG IKXG C NKUV QH UJQTV CPF WUGHWN VKRU

E"VUPNBUJDF 3FTIBQJOH

6Q EJCPIG VJG FKOGPUKQPU QH CP CTTC[[QW ECP QOKV QPG QH VJG UK\GU

```
   B  OQBSBOHF
   C  BSFTIBQF              NFBOT XIBUFWFS JT OFFEFE
   CTIBQF

   C
BSSBZ <<<          >
       <           >
       <           >
       <           >
       <           >>

      <<           >
       <           >
       <           >
       <           >
       <           >>>
```

7FDUPS 4UBDLJOH

*QY FQ YG EQPUVTWEV C & CTTC[HTQO C NKUV QH GSWCNN[UK\GF TQY XGEVQTU? ... VYQ XGEVQTU QH VJG UCOG NGPIVJ ... ETUBDL ... FGRGPFKPI QP VJG FKOGPUKQP KP YJKEJ VJG UVCEMKPI

```
Y  OQBSBOHF
Z  OQBSBOHF
N  OQWTUBDL <Y  Z>
N
BSSBZ <<          >
  <          >>
YZ  OQITUBDL <Y  Z>
YZ
BSSBZ <                    >
```

6JG NQIKE DGJKPF VJQUG HWPEVKQPU KP OQTG VJCP VYQ FKOGPUKQPU ECP

4FF BMTP

/VN1Z GPS ."5-"# VTFST

)JTUPHSBNT

6JG 0WO2[IJTUPHSBN HWPEVKQP CRRNKGF VQ CP CTTC[TGVWTPU C RCKT QH XGEVQ... QH VJG DKP GFNGBU ... HTQO VJG QPG KP 0WO2[6JG ... IJTUPHSBN ... IGPGTCVGU VJG FCVC

```
JNQPSU OVNQZ BT OQ
SH  OQSBOEPN EFGBVMU@SOH
JNQPSU NBUQMPUMJC QZQMPU BT QMU
  #VJME B WFDUPS PG      OPSNBM EFWJBUFT XJUI WBSJBODF    ?  BOE NFB
NV  TJHNB
W  SHOPSNBM NV  TJHNB
  1MPU B OPSNBMJ[FE IJTUPHSBN XJUI    CJOT
QMU IJTU W  CJOT EFOTJUZ 5SVF       NBUQMPUMJC WFSTJPO  QMPU
BSSBZ
  $PNQVUF UIF IJTUPHSBN XJUI OVNQZ BOE UIFO QMPU JU
 O  CJOT  OQIJTUPHSBN W  CJOT EFOTJUZ 5SVF   /VN1Z WFSTJPO  OP QMPU
QMU QMPU    CJOT<    CJOT<> O
```

9KVJ /CVRNQVNKD QT [QW ECP KPURGEV VJG ...

'VSUIFS SFBEJOH

- 6JG 2[VJQP VWVQTKCN
- TGHGTGPEG
- 5EK2[6WVQTKCN
- 5EK2[.GEVWTG 0QVGU
- # OCVNCD 4 +&. 0WO2[5EK2[FKEVKQPCT[
- VWVQTKCN UXF

/6.1: 5)& "#40-65& #"4*$4 '03 #&(*//&34

9GNEQOG VQ VJG CDUQNWVG DGIKPPGTPU IWKFG VQ 0WO2[+H [QW CTG CNTGCF[... IXG EQ
QWV

8FMDPNF UP /VN1Z

0WO2[/VNFSJDBM 1ZUIPO KU C PQRGP UQWTEG 2[VJQP NKDTCT[VJCV PU WUGF KP CNOQ
+VPU VJG WPKXGTUCN UVCPFCTF HQT YQTMKPI YKVJ PWOGTKECN FCVC KP 2
2[&CVC GEQU[UVGOU 0WO2[WUGTU KPENWFG GXGT[QPG HTQO DGIKPPKPI E
CTV UEKGPVK·E CPF KPFWUVTKCN TGUGCTEJ CPF FGXGNQROGPV 6JG 0WO2
UEKMKV NGCTP UEKMKV KOCIG CPF OQUV QVJGT FCVC UEKGPEG CPF UEKGI

6JG 0WO2[NKDTCT[EQPVCKPU OWNVKFKOGPUKQPCN CTTC[CPF OCVTKZ FC
KP NCVGT UGEVKQPU CBZ RTQXKFGU FKOGPUKQPCN CTTC[QDLGEV YKV
KV 0WO2[ECP DG WUGF VQ RGTHQTO C YKFG XCTKGV[QH OCVJGOCVKECN Q
VQ 2[VJQP VJCV IWCTCPVGG G¹EKGPV ECNEWNCVKQPU YKVJ CTTC[U CPF OC
OCVJGOCVKECN HWPEVKQPU VJCV QRGTCVG QP VJGUG CTTC[U CPF OCVTKE

.GCTP OQTG CDQWV /VN1Z IFSF

*OTUBMMJOH /VN1Z

6Q KPUVCNN 0WO2[YG UVTQPIN[TGEQOOGPF WUKPI C UEKGPVK·E 2[VJQP FK
HQT KPUVCNNKPI 0WO2[QP [QWT QRGTCVKPI U[UVGO UGG +PUVCNNKPI 0WO2[

+H [QW CNTGCF[JCXG 2[VJQP [QW ECP KPUVCNN 0WO2[YKVJ

```
DPOEB JOTUBMM OVNQZ
```

QT

```
QJQ JOTUBMM OVNQZ
```

+H [QW FQPPV JCXG 2[VJQP [GV [QW OKIJV YCPV VQ EQPUKFGT WUKPI #PCEQPFC ... UV
VJKPI CDQWV IGVVKPI VJKU FKUVTKDWVKQP KU VJG HCEV VJCV [QW FQPPV P
QT CP[QH VJG OCLQT RCEMCIGU VJCV [QWPNN DG WUKPI HQT [QWT FCVC CPC

)PX UP JNQPSU /VN1Z

6Q CEEGUU 0WO2[CPF KVU HWPEVKQPU KORQTV KV KP [QWT 2[VJQP EQFG NK

```
JNQPSUVN QBZTOQ
```

9G UJQTVGP VJG KORQTV UVCVGOGPV VQ
VJCV [QW UJQWNF HQNNQY UQ VJCV CP[QPG YQTMKPI YKVJ [QWT EQFG ECP G

3FBEJOH UIF FYBNQMF DPEF

+H [QW CTGPPV CNTGCF[EQOHQTVCDNG YKVJ TGCFKPI VWVQTKCNU VJCV EQ
EQFG DNQEM VJCV NQQMU NKMG VJKU

```
B  OQBSBOHF
B   B<OQQFXBYJT  >
B TIBQF
```

+H [QW CTGPPV HCOKNKCT YKVJ VJKU UV[NG
VJCV [QW YQWNF GPVGT
KU VJG UV[NG [QW
UV[NG 0QVG VJCV KV KU PQV RCTV QH VJG EQFG CPF YKNN ECWUG CP GTTQT

8IBUBT UIF EJ FSFODF CFUXFFO B 1ZUIPO MJTU BC

0WO2[IKXGU [QW CP GPQTOQWU TCPIG QH HCUV CPF G¹EKGPV YC[U QH ETGCV
VJGO 9JKNG C 2[VJQP NKUV ECP EQPVCKP FK¶GTGPV FCVC V[RGU YKVJKP C U
DG JQOOQIGPGQWU 6JG OCVJGOCVKECN QRGTCVKQPU VJCV CTG OGCPV VQ D
KH VJG CTTC[U YGTGPPV JQOOQIGPGQWU

8IZ VTF /VN1Z

0WO2[CTTC[U CTG HCUVGT CPF OQTG EQORCEV VJCP 2[VJQP NKUVU #P CTT
0WO2[WUGU OWEJ NGUU OGOQT[VQ UVQTG FCVC CPF KV RTQXKFGU C OGEJC
EQFG VQ DG QRVKOK\GF GXGP HWTVJGT

8IBU JT BO BSSBZ

#P CTTC[KU C EGPVTCN FCVC UVTWEVWTG QH VJG 0WO2[NKDTCT[#P CTTC[
VJG TCY FCVC JQY VQ NQECVG CP GNGOGPV CPF JQY VQ KPVGTRTGV CP GNC
WBSJPVT6JGZT NGOGPVU CTG CNN QH VJG UECOZQVRG TGHGTTGF VQ CU VJG C

#P CTTC[ECP DG KPFGZGF D[C VWRNG QH PQPPGICVKXG KPVGIGTU IKXKP
VJG CTTC[KU VJG PWODGT
GCEJ FKOGPUKQP

1PG YC[YG ECP KPKVKCNK\G 0WO2[CTTC[U KU HTQO 2[VJQP NKUVU WUKPI P

(QT GZCORNG

```
   B   OQBSSBZ <            >
```

QT

```
   B   OQBSSBZ <<      > <         > <           >>
```

9G ECP CEEGUU VJG GNGOGPVU KP VJG CTTC[WUKPI USWCTG DTCEMGVU 9J
KP 0WO2[UVCTVU CV 6JCV OGCPU VJCV KH [QW YCPV VQ CEEGUU VJG ·TUV
S T

```
   QSJOBU<>
<      >
```

.PSF JOGPSNBUJPO BCPVU BSSBZT

5IJT TFDUJPO BOBMZ[FT BSSBZ EBUB TUSVDUVSFT

;QW OKIJV QEECUKQPCNN[JGCT CP CTTC[TGHGTTGF VQ CU C SPFCTTC[T YJ
FKOGPUKQPCN CTTC[KU UKORN[CP CTTC[YKVJ CP[VQ
 % QT VYQ FKOGPUKQPCN CTTC
WFDKUPSP CTTC[YKVJ C UKPING FKOGPUKQP VJGTGPU PQ
VQ CP CTTC[YKVJ VYQ

8IBU BSF UIF BUUSJCVUFT PG BO BSSBZ

#P CTTC[KU WUWCNN[C ·ZGF UK\G EQPVCKPGT QH KVGOU QH VJG UCOG V[R
CP CTTC[KU FG·PGF D[KVU UJCRG 6JG UJCRG QH CP CTTC[KU C VWRNG QH
FKOGPUKQP

+P 0WO2[FKOGPUK

```
<<         >
 <         >>
```

;QWT CTTC[JCU CZGU 6JG ·TUV CZKU JCU C NGPIVJ QH CPF VJG UGEQPF C

,WUV NKMG KP QVJGT 2[VJQP EQPVCKPGT QDLGEVU VJG EQPVGPVU QH CP C
VJG CTTC[7PNKMG VJG V[RKECN EQPVCKPGT QDLGEVU FK¶GTGPV CTTC[U
OKIJV DG XKUKDNG KP CPQVJGT

#TTBZ UUSJTCG¸UGFV KPHQTOCVKQP KPVTKPUKE VQ VJG CTTC[KVUGNH +H [QW
YKVJQWV ETGCVKPI C PGY CTTC[[QW ECP QHVGP CEEGUU CP CTTC[VJTQWI

4GCF OQTG CDQWV CTTC[CVVTKDWVGU JGTG CPF NGCTP CDQWV CTTC[QDLG

)PX UP DSFBUF B CBTJD BSSBZ

5IJT TFDUJPO DPWFST OQ BSSBZ OQ [FSPT OQ POFT OQ FNQUZ OQ BSBOHF OQ MJOTQBDF EUZQF

6Q ETGCVG C 0WO2[CTTC[[QW ECP WUG VJG HWPEVKQP

#NN [QW PGGF VQ FQ VQ ETGCVG C UKORNG CTTC[KU RCUU C NKUV VQ KV +H [QWT NKUV ;QW ECP ·PF OQTG KPHQTOCVKQP CDQWV FCVC V[RGU JGTG

```
JNQPSU OVNQZ BT OQ
B  OQ BSSBZ <  >
```

;QW ECP XKUWCNK\G [QWT CTTC[VJKU YC[

#F BXBSF UIBU UIFTF WJTVBMJ[BUJPOT BSF NFBOU UP TJNQMJGZ JEFBT BOE NFDIBOJDT "SSBZT BOE BSSBZ PQFSBUJPOT BSF NVDI NPSF DPNQMJDBUFE

$GUKFGU ETGCVKPI CP CTTC[HTQO C UGSWGPEG QH GNGOGPVU [QW ECP G

```
OQ [FSPT
BSSBZ <  >
```

1T CP CTTC[·NNGF YKVJ

```
OQ POFT
BSSBZ <  >
```

1T GXGP CP GORV[CTTC[6JG HWPEVKQP ETGCVGU CP CTTC[YJQUG KPKVKCN EQPVGPV KU T QH VJG OGOQT[6JG TGCUQP VQ WUG QXGT UKOKNCT KU URGGF LWUV GNGOGPV CHVGTYCTFU

```
$SFBUF BO FNQUZ BSSBZ XJUI  FMFNFOUT
OQ FNQUZ
BSSBZ <  >  NBZ WBSZ
```

;QW ECP ETGCVG CP CTTC[YKVJ C TCPIG QH GNGOGPVU

```
OQ BSBOHF
BSSBZ <  >
```

#PF GXGP CP CTTC[VJCV EQPVCKPU C TCPIG QH GXGPN[URCEGF KPVGTXCN OVNCFS PG TUFQ TJ[F

```
OQ BSBOHF
BSSBZ <  >
```

;QW ECP CONQUQMWOUTGQBVDQFETGCVG CP CTTC[YKVJ XCNWGU VJCV CTG URCEGF N

```
   OQMJOTQBDF   OVN
BSSBZ <                    >
```

4QFDJGZJOH ZPVS EBUB UZQF

9JKNG VJG FGHCWNV FCVCQV[GRGPKBU[QWCVKPIRZQRKPKVEKVN[URGEKH[YJKEJ FCV
VJEGUZQMFG[YQTF

```
   Y  OQPOFT  EUZQOFQJOU
   Y
BSSBZ <   >
```

-FBSO NPSF BCPVU DSFBUJOH BSSBZT IFSF

"EEJOH SFNPWJOH BOE TPSUJOH FMFNFOUT

5IJT TFDUJOPO DPWFST DPODBUFOBUF

5QTVKPICP GNGOGPQV KEUJQWNEGCYPKUVRJGEKH[VJG CZKU MKPF CPF QTFGT YJ
+H[QW UVCTV YKVJ VJKU CTTC[

```
   BSS OQBSSBZ <                >
```

;QW ECP SWKEMN[UQTV VJG PWODGTU KP CUEGPFKPI QTFGT YKVJ

```
   OQTPSU BSS
BSSBZ <                  >
```

+P CFFKVKQP VQ UQTV YJKEJ TGVWTPU C UQTVGF EQR[QH CP CTTC[[QW EC
YBSHTPSYUJKEJ KU CP KPFKTGEV UQTV CNQPIC URGEK·GF CZKU
YMFYTPSYUJKEJ KU CP KPFKTGEV UVCDNG UQTV QP OWNVKRNG MG[U
YTFBSDITPSYUJKEJ YKNN·PF GNGOGPVU KP C UQTVGF CTTC[CPF
YQBSUJUJYPJCKEJ KU C RCTVKCN UQTV
6Q TGCF OQTG CDQWV UTQPTSVUKPICP CTTC[UGG
+H[QW UVCTV YKVJ VJGUG CTTC[U

```
   B  OQBSSBZ <      >
   C  OQBSSBZ <      >
```

;QW ECP EQPECVGOPQCVDCPVJDBOFKBU F

```
   OQDPODBUFOBUF  B  C
BSSBZ <                  >
```

1T KH[QW UVCTV YKVJ VJGUG CTTC[U

```
   Y  OQBSSBZ <<>  <   >>
   Z  OQBSSBZ <<>>
```

;QW ECP EQPECVGPCVG VJGO YKVJ

```
   OQDPODBUFOBUF  Y  Z   BYJT
BSSBZ <<    >
      <    >
      <    >>
```

+P QTFGT VQ TGOQXG GNGOGPVU HTQO CP CTTC[KVPU UKORNG VQ WUG KPF

6Q TGCF OQTG CDQWV

)PX EP ZPV LOPX UIF TIBQF BOE TJ[F PG BO BSSBZ

5IJT TFDUJPO

OEBSSBZ

OEBSSBZ

OEBSSBZ

(QT GZCORNG KH [QW ETGCVG VJKU CTTC[

```
   BSSBZ@FYBNQQBFSSBZ <<<      >
                       <        >>

                      <<        >
                       <        >>

                      <<        >
                       <        >>>
```

6Q ·PF VJG PWODGT QH FKOGPUKQPU QH VJG CTTC[TWP

```
   BSSBZ@FYBONEJNF
```

6Q ·PF VJG VQVCN PWODGT QH GNGOGPVU KP VJG CTTC[TWP

```
   BSSBZ@FYBTNJQFMF
```

#PF VQ ·PF VJG UJCRG QH [QWT CTTC[TWP

```
   BSSBZ@FYBTNQQF
```

$BO ZPV SFTIBQF BO BSSBZ

5IJT TFDUJPO DPWFST BSS.SFTIBQF

:FT

7UKPI SFTIBQF YKNN IKXG C PGY UJCRG VQ CP CTTC[YKVJQWV EJCPIKPI VJG F
VJG TGUJCRG OGVJQF VJG CTTC[[QW YCPV VQ RTQFWEG PGGFU VQ JCXG VJ
[QW UVCTV YKVJ CP CTTC[YKVJ GNGOGPVU [QW PNN PGGF VQ OCMG UWTG

+H [QW UVCTV YKVJ VJKU CTTC[

```
B OQ BSBOHF
QSJOU B
<                >
```

;QW ECP WUG TGUJCRG VQ TGUJCRG [QWT CTTC[(QT GZCORNG [QW ECP TGUJCRG VJ
VYQ EQNWOPU

```
C B SFTIBQF
QSJOU C
<<   >
 <   >
 <   >>
```

9KVJ SFTIBQF [QW ECP URGEKH[C HGY QRVKQPCN RCTCOGVGTU

```
OQ SFTIBQF B OFXTIBQF PSEFS $
BSSBZ <<                 >>
```

BKU VJG CTTC[VQ DG TGUJCRGF

OFXTIBQF VJG PGY UJCRG [QW YCPV ;QW ECP URGEKH[CP KPVGIGT QT C VWI
TGUWNV YKNN DG CP CTTC[QH VJCV NGPIVJ 6JG UJCRG UJQWNF DG EQORCV

PSEFS $ OGCPU VQ TGCF YTKVG VJG GNGOGPVU WUKPI %-NKMG KPFGZ QTFGT
NKMG KPFGZ QTFGT VQ TGCF YTKVG VJG GNGOGPVU KP (QTVTCP NKMG KPFGZ
% NKMG QTFGT QVJGTYKUG 6JKU KU CP QRVKQPCN RCTCOGVGT CPF FQGU P

+H [QW YCPV VQ NGCTP OQTG CDQWV % CPF (QTVTCP QTFGT [QW ECP TGCF
IFSF UUGPVKCNN[% CPF (QTVTCP QTFGTU JCXG VQ FQ YKVJ JQY KPFKEGU EQ
+P (QTVTCP YJGP OQXKPI VJTQWIJ VJG GNGOGPVU QH C VYQ FKOGPUKQPCN
OQUV TCRKFN[XCT[KPI KPFGZ #U VJG ·TUV KPFGZ OQXGU VQ VJG PGZV TQY
VKOG 6JKU KU YJ[(QTVTCP KU VJQWIJV QH CU C %QNWOP OCLQT NCPIWCIG
OQUV TCRKFN[6JG OCVTKZ KU UVQTGF [EQNWOPU 6JKU KU YJ[% KU CNUQ
QP YJGVJGT KVPU OQTG KORQTVCPV VQ RTGUGTXG VJG KPFGZKPI EQPXGPVK

-FBSO NPSF BCPVU TIBQF NBOJQVMBUJPO IFSF

)PX UP DPOWFSU B % BSSBZ JOUP B % BSSBZ IPX UP BEE B OFX BYJT UP BO BSSBZ

5IJT TFDUJPO DPWFST OQ OFXBYJT FYQBOE@EJNT

;QW ECP WUG PR OGXBYKU VQ KPETGCUG VJG FKOGPUKQPU QH [QWT GZKUVKPI CTTC[

7UKPI PR OFXBYKU YKNN KPETGCUG VJG FKOGPUKQPU QH [QWT CTTC[D[QPG FKOGPUKQP YJGP WUGF QPEG
%CTTC[YKNN DGEQOG %CTTC[%CTTC[YKNN DGEQOG %CTTC[CPF UQ QP

(QT GZCORNG KH [QW UVCTV YKVJ VJKU CTTC[

```
B  OQBSSBZ <          >
B TIBQF
```

;QW ECP WUG PR OGXBYKU VQ CFF C PGY CZKU

```
B  B<OQ OFXBYJT  >
B  TIBQF
```

;QW ECP GZRNKEKVN[EQPXGTV C & CTTC[YKVJ GKVJGT C TQY XGEVQT QT C EQNWOP XGEVQT
[QW ECP EQPXGTV C & CTTC[VQ C TQY XGEVQT D[KPUGTVKPI CP CZKU CNQPI

```
SPX@WFDUPS  B<OQ OFXBYJT  >
SPX@WFDUPS TIBQF
```

1T HQT C EQNWOP XGEVQT [QW ECP KPUGTV CP CZKU CNQPI VJG UGEQPF FKOGPUKQP

```
DPM@WFDUPS  B< OQ OFXBYJT>
DPM@WFDUPS TIBQF
```

;QW ECP CNUQ GZRCPF CP CTTC[D[KPUGTVKPI C PGY CZKU CV C URGEKHKGF RQUKVKQP YKVJ PR GZRCPF@FKOU

(QT GZCORNG KH [QW UVCTV YKVJ VJKU CTTC[

```
B  OQBSSBZ <          >
B TIBQF
```

;QW ECP WUG PR GZRCPF@FKOU VQ CFF CP CZKU CV KPFGZ RQUKVKQP YKVJ

```
C  OQFYQBOE@EJNT B  BYJT
C TIBQF
```

;QW ECP CFF CP CZKU CV KPFGZ RQUKVKQP YKVJ

```
D  OQFYQBOE@EJNT B  BYJT
D TIBQF
```

(KPF OQTG KPHQTOCVKQP CDQWV PGYCZKU JGTG CPF GZRCPF@FKOU CV GZRCPF@FKOU

*OEFYJOH BOE TMJDJOH

;QW ECP KPFGZ CPF UNKEG 0WO2[CTTC[U KP VJG UCOG YC[U [QW ECP UNKEG

```
EBUB OQBSSBZ <     >
EBUB<
EBUB<>
BSSBZ <    >
EBUB<>
BSSBZ <    >
EBUB<>
BSSBZ <    >
```

;QW ECP XKUWCNK\G KV VJKU YC[

;QW OC[YCPV VQ VCMG C UGEVKQP QH [QWT CTTC[QT URGEK·E CTTC[GNGOGP
6Q FQ VJCV [QWPNN PGGF VQ UWDUGV UNKEG CPF QT KPFGZ [QWT CTTC[U

+H [QW YCPV VQ UGNGEV XCNWGU HTQO [QWT CTTC[VJCV HWN·NN EGTVCKP E

(QT GZCORNG KH [QW UVCTV YKVJ VJKU CTTC[

```
B  OQBSSBZ <<        >  <          >  <              >>
```

;QW ECP GCUKN[RTKPV CNN QH VJG XCNWGU KP VJG CTTC[VJCV CTG NGUU V

```
QSJOU B<B   >
<        >
```

;QW ECP CNUQ UGNGEV HQT GZCORNG PWODGTU VJCV CTG GSWCN VQ QT IT

```
GJWF@VQ B
QSJOU B<GJWF@VQ>
<                           >
```

;QW ECP UGNGEV GNGOGPVU VJCV CTG FKXKUKDNG D[

```
EJWJTJCMF@CZ<B    >
QSJOU EJWJTJCMF@CZ@
<                   >
```

1T [QW ECP UGNGEV GNGOGPVU VJCV UCVKUH[VYQ EQPFKVKQPU WUKPI VJG

```
D B< B B >
QSJOU D
< >
```

;QW ECP CNUQ OCMG WUG QH KPFKEGU QT KFKEGU ... CTTC[U VQ DQ QNGCP XCNWGU VJC
XCNWGU KP CP CTTC[HWN·NN C EGTVCKP EQPFKVKQP 6JKU ECP DG WUGHWN

```
GJWF@VQ B ] B
QSJOU GJWF@VQ
<<'BMTF 'BMTF 'BMTF 'BMTF>
 < 5SVF 5SVF 5SVF 5SVF>
 < 5SVF 5SVF 5SVF 5SVF>>
```

;QW ECP CNUQ WUG SVQ UGNGEV GNGOGPVU QT KPFKEGU HTQO CP CTTC[

5VCTVKPI YKVJ VJKU CTTC[

```
B OQBSSBZ << > < > < >>
```

;QW ECP WUG OQO[FSP VQ RTKPV VJG KPFKEGU QH GNGOGPVU VJCV CTG HQT GZC

```
C OQOPO[FSP B
QSJOU C
 BSSBZ < > BSSBZ < >
```

+P VJKU GZCORNG C VWRNG QH CTTC[U YCU TGVWTPGF QPG HQT GCEJ FKOG
VJGUG XCNWGU CTG HQWPF CPF VJG UGEQPF CTTC[TGRTGUGPVU VJG EQNW

+H [QW YCPV VQ IGPGTCVG C NKUV QH EQQTFKPCVGU YJGTG VJG GNGOGPVU
EQQTFKPCVGU CPF RTKPV VJGO (QT GZCORNG

```
MJTU@PG@DPPSEJOBUFT MJTU[JQ C<> C<>

GPS DPPSE JO MJTU@PG@DPPSEJOBUFT
    QSJOU DPPSE
```

;QW ECP CNUQ WUG SVQ RTKPV VJG GNGOGPVU KP CP CTTC[VJCV CTG NGUU V

```
QSJOU B<C>
< >
```

+H VJG GNGOGPV [QW'TG NQQMKPI HQT FQGUPPV GZKUV KP VJG CTTC[VJGP

```
OPU@UIFSF OQOPO[FSP B
QSJOU OPU@UIFSF
 BSSBZ <> EUZQF JOU BSSBZ <> EUZQF JOU
```

.GCTP OQTG CDQWV JOEFYJOH BOE TMJDJOH IFSF

4GCF OQTG CDQWV WUKPI VJG OQO[FSP HWPEVKQP CV

)PX UP DSFBUF BO BSSBZ GSPN FYJTUJOH EBUB

5IJT TFDUJPO JODMVEFT IPX UP DSFBUF BO BSSBZ GSPN FYJTUJOH EBUB JODMVEJOH IPX UP TQMJU JU WJFX BOE DPQZ

;QW ECP GCUKN[ETGCVG C PGY CTTC[HTQO C UGEVKQP QH CP GZKUVKPI CTTC[

.GVPU UC[[QW JCXG VJKU CTTC[

```
B OQBSSBZ < >
```

;QW ECP ETGCVG C PGY CTTC[HTQO C UGEVKQP QH [QWT CTTC[CP[VKOG D[U

```
BSS B< >
BSS
BSSBZ < >
```

*GTG [QW ITCDDGF C UGEVKQP QH [QWT CTTC[HTQO KPFGZ RQUKVKQP VJTQ

;QW ECP CNUQ UVCEM VYQ GZKUVKPI CTTC[U DQVJ XGTVKECNN[CPF JQTK\QPVCNN[

```
B OQBSSBZ << >
< >>

B OQBSSBZ << >
< >>
```

;QW ECP UVCEM VJGO XGTVKECNN[YKVJ

```
OQWTUBDL B B
BSSBZ << >
< >
< >
< >>
```

1T UVCEM VJGO JQTK\QPVCNN[YKVJ

```
OQITUBDL B B
BSSBZ << >
< >>
```

;QW ECP URNKV CP CTTC[KPVQ UGXGTCN UOCNNGT CTTC[U WUKPI PWODGT QH GSWCNN[UJCRGF CTTC[U VQ TGVWTP QT VJG EQNWOPU CHVGT YJKEJ VJG FKXKUKQP UJQWNF QEEWT

.GVPU UC[[QW JCXG VJKU CTTC[

```
Y OQBSBOHF SFTIBQF
Y
BSSBZ << >
< >>
```

+H [QW YCPVGF VQ URNKV VJKU CTTC[KPVQ VJTGG GSWCNN[UJCRGF CTTC[U

```
OQITQMJU Y
<BSSBZ << >
< >> BSSBZ << >
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

```
        <            >>   BSSBZ <<           >
        <            >> >
```

+H [QW YCPVGF VQ URNKV [QWT CTTC[CHVGT VJG VJKTF CPF HQWTVJ EQNWO

```
    OQITQMJU Y
  <BSSBZ <<         >
        <        >>   BSSBZ <<  >
        <  >>   BSSBZ <<                        >
        <                         >> >
```

-FBSO NPSF BCPVU TUBDLJOH BOE TQMJUUJOH BSSBZT IFSF

;QW ECP WUG XKGY VQ ETGCVG C PGY CTTC[QDLGEV VJCV NQQMU CV VJG U DPQZ

8KGYU CTG CP KORQTVCPV 0WO2[EQPEGRV 0WO2[HWPEVKQPU CU YGNN C XKGYU YJGPGXGT RQUUKDNG 6JKU UCXGU OGOQT[CPF KU HCUVGT PQ EQR VQ DG CYCTG QH VJKU OQFKH[KPI FCVC KP C XKGY CNUQ OQFK·GU VJG QTKI

.GVPU UC[[QW ETGCVG VJKU CTTC[

```
    B   OQBSSBZ <<      >  <         >  <           >>
```

0QY YG ETGCVG CP CTTC[D D[UNKEKPI C CPF OQFKH[VJG · TUV GNGOGPV QH D 6JKU YKNN OQFKH[VJG EQTTGURQ KB CU YGNN

```
    C    B<   >
    C
BSSBZ <          >
    C <>
    C
BSSBZ <             >
    B
BSSBZ <<              >
      <              >
      <              >>
```

7UKPI VJG EQR[OGVJQF YKNN OCMG C EQORNGVG EQR[QH VJG CTTC[CPF KVU FCVC EQWNF TWP

```
    C    B DPQZ
```

-FBSO NPSF BCPVU DPQJFT BOE WJFXT IFSF

#BTJD BSSBZ PQFSBUJPOT

5IJT TFDUJPO DPWFST BEEJUJPO TVCUSBDUJPO NVMUJQMJDBUJPO EJWJT

1PEG [QWPXG ETGCVGF [QWT CTTC[U [QW ECP UVCTV VQ YQTM YKVJ VJGO .G QPG ECNNGF SFCVCT CPF QPG ECNNGF SQPGUT

;QW ECP CFF VJG CTTC[U VQIGVJGT YKVJ VJG RNWU UKIP

```
    EBUB OQBSSBZ < >
    POFT OQPOFT EUZQJFOU
    EBUB POFT
BSSBZ <    >
```

;QW ECP QH EQWTUG FQ OQTG VJCP LWUV CFFKVKQP

```
    EBUB POFT
BSSBZ <    >
    EBUB EBUB
BSSBZ <    >
    EBUB EBUB
BSSBZ <      >
```

$CUKE QRGTCVKQPU CTG UKORNG YKVJ 0WO2[+H [QW YCPVTVVQ · BFKVJG UWO
YQTMU HQT &CTTC[U &CTTC[U CPF CTTC[U KP JKIJGT FKOGPUKQPU

```
    B  OQBSSBZ <       >

    B TVN
```

6Q CFF VJG TQYU QT VJG EQNWOPU KP C &CTTC[[QW YQWNF URGEKH[VJG C
+H [QW UVCTV YKVJ VJKU CTTC[

```
C OQBSSBZ <<> < >>
```

;QW ECP UWO QXGT VJG CZKU QH TQYU YKVJ

```
C TVN BYJT
BSSBZ < >
```

;QW ECP UWO QXGT VJG CZKU QH EQNWOPU YKVJ

```
C TVN BYJT
BSSBZ < >
```

-FBSO NPSF BCPVU CBTJD PQFSBUJPOT IFSF

#SPBEDBTUJOH

6JGTG CTG VKOGU YJGP [QW OKIJV YCPV VQ ECTT[QWV CP QRGTCVKQP DGV PQFSBUJPO CFUXFFO BO WFDUPS BOE B TDBMBS VYQ FK¶GTGPV UK\GU (QT G SFCVCT OKIJV EQPVCKP KPHQTOCVKQP CDQWV FKUVCPEG KP OKNGU DWV [Q RGTHQTO VJKU QRGTCVKQP YKVJ

```
EBUB OQBSSBZ < >
EBUB
BSSBZ < >
```

0WO2[WPFGTUVCPFU VJCV VJG OWNVKRNKECVKQP UJQWNF JCRRGP YKVJ G ECUVKPI KU C OGEJCPKUO VJCV CNNQYU 0WO2[VQ RGTHQTO QRGTCVKQPU Q CTTC[OWUV DG EQORCVKDNG HQT GZCORNG YJGP VJG FKOGPUKQPU QH DQ FKOGPUKQPU CTG PQV EQORCVKDNG [QW YKNN IGV C

-FBSO NPSF BCPVU CSPBEDBTUJOH IFSF

.PSF VTFGVM BSSBZ PQFSBUJPOT

5IJT TFDUJPO DPWFST NBYJNVN NJOJNVN TVN NFBO QSPEVDU TUBOEBSE

0WO2[CNUQ RGTHQTOU CIITGICVKQP HWPEVKQPU KP CFFKVKQP VQ OKP OCZ CXGTCIG IGV VJG TGUWNV QH OWNVKRN[KPI VJG GNGOGPVU VQIGVJGT

```
    EBUBNBY

    EBUBNJO

    EBUBTVN
```

.GVPU UVCTV YKVJ VJKU CTTC[ECNNGF SCT

```
    B  OQBSSBZ <<                                    >
               <                                      >
               <                                      >>
```

+VPU XGT[EQOOQP VQ YCPV VQ CIITGICVG CNQPI C TQY QT EQNWOP $[FGHC
VJG CIITGICVG QH VJG GPVKTG CTTC[6Q·PF VJG UWO QT VJG OKPKOWO QH V

```
    B TVN
```

1T

```
    B NJO
```

;QW ECP URGEKH[QP YJKEJ CZKU [QW YCPV VJG CIITGICVKQP HWPEVKQP VQ D
XCNWG YKVJKP GCEJ EBQYNJWOP D[URGEKH[KPI

```
    B NJO BYJT
BSSBZ <                                  >
```

6JG HQWT XCNWGU NKUVGF CDQXG EQTTGURQPF VQ VJG PWODGT QH EQNWO
HQWT XCNWGU CU [QWT TGUWNV

4GCF OQTG CDQWV CTTC[OGVJQFU JGTG

$SFBUJOH NBUSJDFT

;QW ECP RCUU 2[VJQP NKUVU QH NKUVU VQ ETGCVG C &CTTC[QT SOCVTKZ

```
    EBUB OQBSSBZ <<>  <   >  <   >>
    EBUB
BSSBZ <<   >
      <   >
      <   >>
```

+PFGZKPI CPF UNKEKPI QRGTCVKQPU CTG WUGHWN YJGP [QWPTG OCPKRWN

```
    EBUB< >

    EBUB<>
BSSBZ <<     >
     <    >>
    EBUB<    >
BSSBZ <    >
```

;QW ECP CIITGICVG OCVTKEGU VJG UCOG YC[[QW CIITGICVGF XGEVQTU

```
    EBUBNBY

    EBUBNJO

    EBUBTVN
```

;QW ECP CIITGICVG CNN VJG XCNWGU KP C OCVTKZ CPF [QW BYJT ECP CIITGICVG VJ
6Q KNNWUVTCVG VJKU RQKPV NGVPU NQQM CV C UNKIJVN[OQFK·GF FCVCUG

```
    EBUB OQBSSBZ <<>   <    >  <    >>
    EBUB
BSSBZ <<     >
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

```
      <    >
      <    >>
   EBUBNBY BYJT
BSSBZ <    >
   EBUBNBY BYJT
BSSBZ <      >
```

1PEG [QWPXG ETGCVGF [QWT OCVTKEGU [QW ECP CFF CPF OWNVKRN[VJGO
VJCV CTG VJG UCOG UK\G

```
   EBUB OQBSSBZ <<>  <    >>
   POFT OQBSSBZ <<>  <    >>
   EBUB POFT
BSSBZ <<    >
      <    >>
```

;QW ECP FQ VJGUG CTKVJOGVKE QRGTCVKQPU QP OCVTKEGU QH FK¶GTGPV U
TQY +P VJKU ECUG 0WO2[YKNN WUG KVU DTQCFECUV TWNGU HQT VJG QRGT

```
   EBUB OQBSSBZ <<>  <    >  <    >>
   POFT@SPXQBSSBZ <<>>
   EBUB POFT@SPX
BSSBZ <<    >
      <    >
      <    >>
```

$G CYCTG VJCV YJGP 0WO2[RTKPVU 0 FKOGPUKQPCN CTTC[U VJG NCUV CZK
UNQYGUV (QT KPUVCPEG

```
   OQPOFT
BSSBZ <<<     >
       <      >
       <      >>

      <<      >
       <      >
       <      >>

      <<      >
       <      >
       <      >>

      <<      >
       <      >
       <      >>>
```

6JGTG CTG QHVGP KPUVCPEGU YJGTG YG YCPV 0WO2[VQ KPKVKCVK\G VJG XC
CP[HSPT CPFSVBGEPN (FOFSNCPS HQT TCPFQO PWODGT IGPGTCVKQP HQT VJ
KP VJG PWODGT QH GNGOGPVU [QW YCPV KV VQ IGPGTCVG

```
   OQPOFT
BSSBZ <        >
   OQ[FSPT
BSSBZ <        >
   SOH OQSBOEPNFGBVMU@SOHUIF TJNQMFTU XBZ UP HFOFSBUF SBOEPN OVNCFST
   SOHSBOEPN
BSSBZ <                        >
```

;QW ECP CPNQTWFSGPT CPSFBOEPNVQ ETGCVG C & CTTC[KH [QW IKXG VJGO C
FKOGPUKQPU QH VJG OCVTKZ

```
   OQPOFT
BSSBZ <<     >
      <      >
      <      >>
   OQ[FSPT
BSSBZ <<     >
      <      >
      <      >>
   SOHSBOEPN
BSSBZ <<          >
      <           >
      <           >>     NBZ WBSZ
```

4GCF OQTG CDQWV ETGCVPKUPIUCTQTVCJ[GUT XNCNNGWFGYUKQVJ WPKPKVKCNK\GF CV CTT

(FOFSBUJOH SBOEPN OVNCFST

6JG WUG QH TCPFQO PWODGT IGPGTCVKQP KU CP KORQTVCPV RCTV QH VJG OCEJKPG NGCTPKPI CNIQTKVJOU 9JGVJGT [QW PGGF VQ TCPFQON[KPKVKC KPVQ TCPFQO UGVU QT TCPFQON[UJWºG [QWT FCVCUGV DGKPI CDNG VQ IG TCPFQO PWODGTU KU GUUGPVKCN

9KVJ (FOFSBUPS JOUFHFST [QW ECP IGPGTCVG TCPFQO KPVGIGTU HTQO NQY TGOC 0WO2[VQ JKIJ GZENWUKXG ;QW ECP UGV GPFRQKPV VJG JKIJ PWODGT KPENWUKXG

;QW ECP IGPGTCVG C Z CTTC[QH TCPFQO KPVGIGTU DGVYGGP CPF YKVJ

```
SOH JOUFHFST TJ[F
BSSBZ << >
< >> NBZ WBSZ
```

4GCF OQTG CDQWV TCPFQO PWODGT IGPGTCVKQP JGTG

)PX UP HFU VOJRVF JUFNT BOE DPVOUT

5IJT TFDUJPO DPWFST

;QW ECP ·PF VJG WPKSWG GNGOGPVU KP CP CTTC[GCUKN[YKVJ

(QT GZCORNG KH [QW UVCTV YKVJ VJKU CTTC[

```
B OQBSSBZ < >
```

[QW ECP VUG JRVF RTKPV VJG WPKSWG XCNWGU KP [QWT CTTC[

```
VOJRVF@WBMVFT OQVOJRVF B
QSJOU VOJRVF@WBMVFT
< >
```

6Q IGV VJG KPFKEGU QH WPKSWG XCNWGU KP C 0WO2[CTTC[CP CTTC[QH ·TU RCUU SFUVSO@JOEFY CPF VJG XCNWG YGNN CU [QWT CTTC[

```
VOJRVF@WBMVFT JOEJDFT@MJTU B SFUVSO@JOEFY
QSJOU JOEJDFT@MJTU
< >
```

;QW ECP RCUU SFUVSO@DPVOUT KP PJRVF CNQPI YKVJ [QWT CTTC[VQ IGV VJG HT WPKSWG XCNWGU KP C 0WO2[CTTC[

```
VOJRVF@WBMVFT PDDVSSFODF@DPVOU B SFUVSO@DPVOUT
QSJOU PDDVSSFODF@DPVOU
< >
```

6JKU CNUQ YQTMU YKVJ & CTTC[U +H [QW UVCTV YKVJ VJKU CTTC[

```
B@ E OQBSSBZ << > < > < > < >>
```

;QW ECP ·PF WPKSWG XCNWGU YKVJ

```
VOJRVF@WBMVFT OJRVF B@ E
QSJOU VOJRVF@WBMVFT
<                                >
```

+H VJG CZKU CTIWOGPV KUPPV RCUUGF [QWT &CTTC[YKNN DG ¸CVVGPGF

+H [QW YCPV VQ IGV VJG WPKSWG TQYU BYJT CPF QIWOGPU 6QMP FUWJG WPKSWGUT
BYJT CPF HQT EQNWOPU URGEKH[

```
VOJRVF@SPXT OJRVF B@ E BYJT
QSJOU VOJRVF@SPXT
<<          >
 <          >
 <          >>
```

6Q IGV VJG WPKSWG TQYU KPFGZ RQUKVKQP CPF QEEWTTGPEG EQWPV [QW

```
VOJRVF@SPXT JOEJDFT PDDVSSFODF@DPVOU
    B@ E BYJTSFUVSO@DPVOUTSFUVSO@JOEFY
QSJOU VOJRVF@SPXT
<<          >
 <          >
 <          >>
QSJOU JOEJDFT
<     >
QSJOU PDDVSSFODF@DPVOU
<     >
```

6Q NGCTP OQTG CDQWV ·PFKPI VJG WPKSWG GNGOGPVU KP CP CTTC[UGG

5SBOTQPTJOH BOE SFTIBQJOH B NBUSJY

5IJT TFDUJPO SFWJFXT USBOTQPTF 5

+VPU EQOOQP VQ PGGF VQ VTCPURQUG [QWT OCVTKEGU

;QW OC[CNUQ PGGF VQ UYKVEJ VJG FKOGPUKQPU QH C OCVTKZ 6JKU ECP J
GZRGEVU C EGTVCKP KPRWV UJCRG VJCV KU FKHHGTGPV HTQO
UKORN[PGGF VQ RCUU KP VJG PGY FKOGPUKQPU VJCV [QW YCPV HQT VJG OC

```
   EBUBSFTIBQF
BSSBZ <<      >
      <      >>
   EBUBSFTIBQF
BSSBZ <<    >
      <    >
      <    >>
```

;QW ECP CNUSQWTQGTVE TGXGTUG QT EJCPIG VJG CZGU QH CP CTTC[CEEQTFK

+H [QW UVCTV YKVJ VJKU CTTC[

```
   BSS OQBSBOHF SFTIBQF
   BSS
BSSBZ <<      >
      <      >>
```

;QW ECP VTCPURQUBSS[QWS ECTTQPYK VJ

```
   BSSUSBOTQPTF
BSSBZ <<    >
      <    >
      <    >>
```

;QW ECP CBNSUSQ5WUG

```
   BSS5
BSSBZ <<    >
      <    >
      <    >>
```

6Q NGCTP OQTG CDQWV VTCPURQUSKPITQRHTGTUBCQFK PI CTTC[U UGG

)PX UP SFWFSTF BO BSSBZ

5IJT TFDUJPO DPWFST

0WO2[QPU GMJQ HWPEVKQP CNNQYU [QW VQ ¸KR QT TGXGTUG VJG QTFGT QH VJG GNGOGPVU QH CP CTTC[CNQPI CP CZKU
GMJQ URGEKH[VJG CTTC[[QW YQWNF NKMG VQ TGXGTUG CPF VJG CZKU +H [QW FQP V URGEKH[VJG CZKU PWOR[TGXGTUGU VJG
EQPVGPVU CNQPI CNN QH VJG CZGU QH [QWT KPRWV CTTC[

3FWFSTJOH B % BSSBZ

+H [QW DGIKP YKVJ C & CTTC[NKMG VJKU QPG

```
BSS OQBSSBZ <                    >
```

;QW ECP TGXGTUG KV YKVJ

```
SFWFSTFE@BSS OQGMJQ BSS
```

+H [QW YCPV VQ RTKPV [QWT TGXGTUGF CTTC[[QW ECP TWP

```
QSJOU 3FWFSTFE "SSBZ  SFWFSTFE@BSS
3FWFSTFE "SSBZ  <                 >
```

3FWFSTJOH B % BSSBZ

& CTTC[YQTMU OWEJ VJG UCOG YC[

+H [QW UVCTV YKVJ VJKU CTTC[

```
BSS@ EOQBSSBZ <<        > <          > <             >>
```

;QW ECP TGXGTUG VJG EQPVGPV KP CNN QH VJG TQYU CPF CNN QH VJG EQNWOPU YKVJ

```
SFWFSTFE@BSS OQGMJQ BSS@ E
QSJOU SFWFSTFE@BSS
<<            >
 <            >
 <            >>
```

;QW ECP GCUKN[TGXGTUG QPN[VJG TQYU YKVJ

```
SFWFSTFE@BSS@SPXT OQGMJQ BSS@ E  BYJT
QSJOU SFWFSTFE@BSS@SPXT
<<            >
 <            >
 <            >>
```

1T TGXGTUG QPN[VJG EQNWOPU YKVJ

```
SFWFSTFE@BSS@DPMVNOT OQGMJQ BSS@ E  BYJT
QSJOU SFWFSTFE@BSS@DPMVNOT
<<            >
 <            >
 <            >>
```

;QW ECP CNUQ TGXGTUG VJG EQPVGPVU QH QPN[QPG EQNWOP QT TQY (QT GZCORNG [QW EQWNF TGXGTUG VJG EQPVGPVU QH VJG TQY CV
KPFGZ RQUKVKQP VJG UGEQPF TQY

```
    BSS@ E< OQGMJQ BSS@ E<
    QSJOU BSS@ E
<<         >
 <         >
 <         >>
```

;QW ECP CNUQ TGXGTUG VJG EQNWOP CV KPFGZ RQUKVKQP VJG UGEQPF EQ

```
    BSS@ E<> OQGMJQ BSS@ E<
    QSJOU BSS@ E
<<         >
 <         >
 <         >>
```

4GCF OQTG CDQWV TGXGTUKPI CTTC[U CV

3FTIBQJOH BOE "BUUFOJOH NVMUJEJNFOTJPOBM

5IJT TFDUJPO DPWFST GMBUUFO SBWFM

6JGTG CTG VYQ RQRWNCT YC[U VQ HNCVVGP CP CTTC[6JGTG CTG VYQ RQRWNCT YC[U VQ HNCVVGP CP CTTC[FK¶GTGPEG DGV VYQ KU VJCV VJG PGY CTTC[ETGCVGF WUKPI TGHGTGPEG VQ VJG RCTGPV CTT VJCV CP[EJCPIGU VQ VJG PGY CTTC[YKNN PQV CHHGEV VJG RCTGPV CTTC[G¹EKGPV

+H [QW UVCTV YKVJ VJKU CTTC[

```
    Y   OQBSSBZ <<       >  <        >  <          >>
```

;QW ECP WUG HNCVVGP VQ HNCVVGP [QWT CTTC[KPVQ C & CTTC[

```
    Y GMBUUFO
BSSBZ <                                  >
```

9JGP [QW WUG HNCVVGP EJCPIGU VQ [QWT PGY CTTC[YQPPV EJCPIG VJG RCTGPV CTTC

(QT GZCORNG

```
    B   Y GMBUUFO
    B <>
    QSJOU Y    0SJHJOBM BSSBZ
<<         >
 <         >
 <         >>
    QSJOU B    /FX BSSBZ
<                            >
```

$WV YJGP [QW WUG TCXGN EJCPIGU [QW OCMG VQ VJG PGY CTTC[YKNN C¶GEV VJG R

(QT GZCORNG

```
    B   Y SBWFM
    B <>
    QSJOU Y    0SJHJOBM BSSBZ
```

EQPVKPWGU QP PGZV RCIG

EQPVKPWGF HTQO RTGXKQWU

```
<<          >
 <          >
 <          >>
   QSJOBU     /FX BSSBZ
<                          >
```

4GCF OQTG GZCORNGU QH VJKU KU VJG DWKNV KP CEEGUU VQ FQEWOGPVCVKQP 'XGT[CU VJG FCVC UEKGPEG GEQU[UVGO 2[VJQP CPF 0WO2[CTG 2[VJQP JCU CFFKVKQPCN PGGF OQTG KPHQTOCVKQP VJCV [QW PGGF VQ FQ

)PX UP BDDFTT UIF EPDTUSJOH GPS NPSF JOGPSN

5IJT TFDUJPO QQWFST

9JGP KV EQOGU VQ VJG FCVC UEKGPEG GEQU[UVGO 2[VJQP CPF 0WO2[CTG GZCORNGU QH VJKU KU VJG DWKNV KP CEEGUU VQ FQEWOGPVCVKQP 'XGT[CU VJG FCVC UEKGPEG GEQU[UVGO 2[VJQP CPF EQPE 2[VJQP JCU CFFKVKQPCN PGGF OQTG KPHQTOCVKQP VJCV ECP JGNR [QW CEEGUU VJKU KPHQTOCVKQ PGGF OQTG KPHQTOCVKQP VJCV [QW PGG

(QT GZCORNG

```
   IFMQNBY
)FMQ PO CVJMU JO GVODUJPO NBY JO NPEVMF CVJMUJOT

NBY
   NBY JUFSBCMF   <  EFGBVMU PCK  LFZ GVOD>     WBMVF
   NBY BSH   BSH    BSHT   <  LFZ GVOD>     WBMVF

   8JUI B TJOHMF JUFSBCMF BSHVNFOU  SFUVSO JUT CJHHFTU JUFN  5IF
   EFGBVMU LFZXPSE POMZ BSHVNFOU TQFDJGJFT BO PCKFDU UP SFUVSO JG
   UIF QSPWJEFE JUFSBCMF JT FNQUZ
   8JUI UXP PS NPSF BSHVNFOUT  SFUVSO UIF MBSHFTU BSHVNFOU
```

$GECWUG CEEGUU VQ CFFKVKQPCN KPHQTOCVKQP VJCV [QW FQEWOGPVCVKQP CNQPI YKVJ QVJGT TGNGXCPV KPHQTOCVKQP +2[VJQP KU NCPIWCIG ECP ·PF OQTG KPHQTOCVKQP CDQWV +2[VJQP JGTG

(QT GZCORNG

```
*O  <  > NBY
NBY JUFSBCMF   <  EFGBVMU PCK  LFZ GVOD>     WBMVF
NBY BSH   BSH    BSHT   <  LFZ GVOD>     WBMVF

8JUI B TJOHMF JUFSBCMF BSHVNFOU  SFUVSO JUT CJHHFTU JUFN  5IF
EFGBVMU LFZXPSE POMZ BSHVNFOU TQFDJGJFT BO PCKFDU UP SFUVSO JG
UIF QSPWJEFE JUFSBCMF JT FNQUZ
8JUI UXP PS NPSF BSHVNFOUT  SFUVSO UIF MBSHFTU BSHVNFOU
5ZQF     CVJMUJO@GVODUJPO@PS@NFUIPE
```

;QW ECP GXGP WUG VJKU PQVCVKQP HQT QDLGEV OGVJQFU CPF QDLGEVU VJ

.GVPU UC[[QW ETGCVG VJKU CTTC[

```
   B   OQBSSBZ <            >
```

6JGP [QW ECP QDVCKP C NQV QH WUGHWN KPHQTOCVKQP FKTGEVN[HTQO VJG FQEUVTKPI QH C HWPEVKQP YJKEJ KU CP KPUVCPEG

```
*O < > B
5ZQF OEBSSBZ
4USJOH GPSN< >
-FOHUI
'JMF _ BOBDPOEB MJC QZUIPO TJUF QBDLBHFT OVNQZ @@JOJU@@ Q
%PDTUSJOH OP EPDTUSJOH
$MBTT EPDTUSJOH
OEBSSBZ TIBQF EUZQF GMPBU CVGGFS /POF PGGTFU
TUSJEFT /POF PSEFS /POF

"O BSSBZ PCKFDU SFQSFTFOUT B NVMUJEJNFOTJPOBM IPNPHFOFPVT BSSBZ
PG GJYFE TJ[F JUFNT "O BTTPDJBUFE EBUB UZQF PCKFDU EFTDSJCFT UIF
GPSNBU PG FBDI FMFNFOU JO UIF BSSBZ JUT CZUF PSEFS IPX NBOZ CZUFT JU
PDDVQJFT JO NFNPSZ XIFUIFS JU JT BO JOUFHFS B GMPBUJOH QPJOU OVNCFS
PS TPNFUIJOH FMTF FUD

"SSBZT TIPVME CF DPOTUSVDUFE VTJOH ABSSBZA A[FSPTA PS AFNQUZA SFGFS
UP UIF 4FF "MTP TFDUJPO CFMPX 5IF QBSBNFUFST HJWFO IFSF SFGFS UP
B MPX MFWFM NFUIPE AOEBSSBZ A GPS JOTUBOUJBUJOH BO BSSBZ

'PS NPSF JOGPSNBUJPO SFGFS UP UIF AOVNQZA NPEVMF BOE FYBNJOF UIF
NFUIPET BOE BUUSJCVUFT PG BO BSSBZ

1BSBNFUFST

GPS UIF @@OFX@@ NFUIPE TFF /PUFT CFMPX

TIBQF UVQMF PG JOUT
4IBQF PG DSFBUFE BSSBZ
```

6JKU CNUQ YQTMU HQT HWPEVKQPU CPF QVJGT QDLGEVU YKVJ C FQEUVTKPI CUUKIPGF KPENWFG C FQE
WUKPI C UVTKPI NKVGTCN CU VJG HKTUV NKPG KPUVCPEG [QW FQEWOGPVCVKQP

(QT GZCORNG KH [QW ETGCVG VJKU HWPEVKQP

```
EFG EPVCMF B
3FUVSO B
SFUVSO B
```

;QW ECP QDVCKP KPHQTOCVKQP CDQWV VJG HWPEVKQP

```
*O < > EPVCMF
4JHOBUVSF EPVCMF B
%PDTUSJOH 3FUVSO B
'JMF _ %FTLUPQ JQZUIPO JOQVU C BEG CF
5ZQF GVODUJPO
```

;QW ECP TGCEJ CPQVJGT NGXGN QH KPHQTOCVKQP D[TGCFKPI VJG UQWTEG EQFG QH VJG HWPEVKQP
SWGUVKQP OCTM CNNQYU [QW VQ CEEGUU VJG UQWTEG EQFG

(QT GZCORNG

```
*O < > EPVCMF
4JHOBUVSF EPVCMF B
```

EQPVKPWGU QP PGZV RCIG

EQPVKPWGF HTQO RTGXKQWU

```
4PVSDF
EFG EPVCMF B
     3FUVSO B
   SFUVSO B
'JMF     _ %FTLUPQ  JQZUIPO JOQVU    C BEG  CF
5ZQF     GVODUJPO
```

+H VJG QDLGEV KP SWGUVKQP KU EQORKNGF YKVJ ... KP C NQV QH DWKNV KP QDLGEVU CPF V[RGU HQT GZCOR
;QWPNN ·PF VJKU YKVJ C NQV QH DWKNV KP QDLGEVU CPF V[RGU HQT GZCOR

```
*O < >MFO
4JHOBUVSF  MFO PCK
%PDTUSJOH  3FUVSO UIF OVNCFS PG JUFNT JO B DPOUBJOFS
5ZQF     CVJMUJO@GVODUJPO@PS@NFUIPE
```

CPF

```
*O < >MFO
4JHOBUVSF  MFO PCK
%PDTUSJOH  3FUVSO UIF OVNCFS PG JUFNT JO B DPOUBJOFS
5ZQF     CVJMUJO@GVODUJPO@PS@NFUIPE
```

JCXG VJG UCOG QWVRWV DGECWUG VJG[YGTG EQORKNGF KP C RTQITCOOKP

8PSLJOH XJUI NBUIFNBUJDBM GPSNVMBT

6JG GCUG QH KORNGOGPVKPI OCVJGOCVKECN HQTOWNCU VJCV YQTM QP CT WUGF KP VJG UEKGPVK·E 2[VJQP EQOOWPKV[

(QT GZCORNG VJKU KU VJG OGCP USWCTG GTTQT HQTOWNC C EGPVTCN HQ FGCN YKVJ TGITGUUKQP

+ORNGOGPVKPI VJKU HQTOWNC KU UKORNG CPF UVTCKIJVHQTYCTF KP 0WO2

9JCV OCMGU VJKU YQTM UQ YGNN KU VJCV ... EQPVCKP QPG QT C VJQWUCPF XC PGGF VQ DG VJG UCOG UK\G

;QW ECP XKUWCNK\G KV VJKU YC[

+P VJKU GZCORNG DQVJ VJG RTGFKEVKQPU CPF [illegible]
ECTT[QWV UWDVTCEVKQPU VJG XCNWGU KP VJG XGEVQT CTG USWCTGF 6J
XCNWG HQT VJCV RTGFKEVKQP CPF C UEQTG HQT VJG SWCNKV[QH VJG OQFG

)PX UP TBWF BOE MPBE /VN1Z PCKFDUT

5IJT TFDUJPO [illegible] TBWF [illegible] MPBE [illegible] MPBEUYU

;QW YKNN CV UQOG RQKPV YCPV VQ UCXG [QWT CTTC[U VQ FKUM CPF NQCF VJ
VJGTG CTG UGXGTCN YC[U VQ UCXG CPF NQCF QDLGEVU YKVJ 0WO2[6JG PF
FKUM [illegible] VJCV JCP [illegible]
0WO2[DKPCT[[illegible] VJCV JCPFNG [illegible]

6JG [illegible] FQ[NGU UVQTG FCVC UJCRG FV[RG CPF QVJGT KPHQTOCVKQP TGS
CNNQYU VJG CTTC[VQ DG EQTTGEVN[TGVTKGXGF GXGP YJGP VJG [illegible]

+H [QW YCPV VQ UVQTG C UKPING PFCTT [illegible]
PFCTTC[QDLGEV KP C UKPING [illegible] CTTC[U
EQORTGUUGF PR [illegible]

+VPU GCU[VQ UCXG CPF [illegible] VQ URGEKH[VJG CTTC[[QV
PCOG (QT GZCORNG KH [QW ETGCVG VJKU CTTC[

B QQBSSBZ < >

;QW ECP UCXG KV CU S·NGPCOG PR[T YKVJ

```
OQTBWFGJMFOBNF B
```

;QW ECP NQCF VJG PEQ TGEQPUVTWEV [QWT CTTC[

```
C OQMPBEGJMFOBNF OQZ
```

+H [QW YCPV VQ EJGEM [QWT CTTC[[QW ECP TWP

```
QSJOU C
<                 >
```

;QW ECP UCXG C 0WO2[CTTC[CU C RNCKP VGZV HKNG NKMG C EUX HKNG YKVJ

(QT GZCORNG KH [QW ETGCVG VJKU CTTC[

```
DTW@BSSBZ OQBSSBZ <                                    >
```

;QW ECP GCUKN[UCXG KV CU C EUX ·NG YKVJ VJG PCOG SPGYA·NG EUXT NKM

```
OQTBWFUYU OFX@GJMF DTW DTW@BSS
```

;QW ECP SWKEMN[CPF GCUKN[NQCF [QWT UCXGF VGZV ·NG WUKPI

```
OQMPBEUYU OFX@GJMF DTW
BSSBZ <                                             >
```

6JG TBWFUYU BOE MPBEUYU GVPEVKQPU CEEGRV CFFKVKQPCN QRVKQPCN RCTCOGV
9JKNG VGZV ·NGU ECP DG GCUKGT HQT UJCTKPI PR[CPF PR\ ·NGU CTG UOCI
JCPFNKPI QH [QWT VGZV ·NG HQT GZCORNG KH [QW PGGF VQ YQTM YKVJ NK
VJG OGSPHUWFUVKQP

9KVJ TBWFUYU [QW ECP URGEKH[JGCFGTU HQQVGTU EQOOGPVU CPF OQTG

.GCTP OQTG CDQWV KPRWV CPF QWVRWV TQWVKPGU JGTG

*NQPSUJOH BOE FYQPSUJOH B $47

+VPU UKORNG VQ TGCF KP C %58 VJCV EQPVCKPU GZKUVKPI KPHQTOCVKQP

```
JNQPSU QBOEBT BT QE

  *G BMM PG ZPVS DPMVNOT BSF UIF TBNF UZQF
Y  QE SFBE@DTW NVTJD DTW  WBMVFT
QSJOU Y
<< #JMMJF )PMJEBZ   +B[[              >
 < +JNNJF )FOESJY   3PDL              >
 < .JMFT %BWJT   +B[[              >
 < 4*"  1PQ                 >>

  :PV DBO BMTP TJNQMZ TFMFDU UIF DPMVNOT ZPV OFFE
Y  QE SFBE@DTW NVTJD DTW VTFDPMT <TSUJTU 1MBZT>  WBMVFT
QSJOU Y
<< #JMMJF )PMJEBZ         >
 < +JNNJF )FOESJY         >
```

EQPVKPWGU QP PGZV RCI

EQPVKPWGF HTQO RTGXKQWU

```
< .JMFT %BWJT          >
< 4*"          >>
```

+VPU UKORNG VQ WUG 2CPFCU KP QTFGT VQ GZRQTV [QWT CTTC[CU YGNN +H
FCVCHTCOG HTQO VJG XCNWGU KP [QWT CTTC[CPF VJGP YTKVG VJG FCVC HT

+H [QW ETGCVGF VJKU CTTC[SCT

```
   B   OQBSSBZ <<                                          >
               <                                           >
               <                                           >
               <                                           >>
```

;QW EQWNF ETGCVG C 2CPFCU FCVCHTCOG

```
   EG  QE%BUB'SBNF B
   QSJOEUG
```

;QW ECP GCUKN[UCXG [QWT FCVCHTCOG YKVJ

```
   EGUP@DTQWE DTW
```

#PF TGCF [QWT %58 YKVJ

```
   EBUB QESFBE@DTQW DTW
```

;QW ECP CNUQ UCXG [QWTTCBTWTFCUYGKWJQVFJG 0WO2[

```
   OQTBWFUYUQ DTWB  GNU  G  EFMJNJUFS IFBEFS
```

+H [QWPTG WUKPI VJG EQOOCPF NKPG [QW ECP TGCF [QWT UCXGF %58 CP[V

```
DBU OQ DTW
```

1T [QW ECP QRGP VJG ·NG CP[VKOG YKVJ C VGZV GFKVQT

+H [QWPTG KPVGTGUVGF KP NGCTPKPI OQTG CDQWV 2CPFCU 2CPFCU YKVJ 2CPFCU KPUVCNNCVKQP KPHQTOCVKQP

1MPUUJOH BSSBZT XJUI .BUQMPUMJC

+H [QW PGGF VQ IGPGTCVG C RNQV HQT [QWT XCNWGU KVPU XGT[UKORNG YKVJ

(QT GZCORNG [QW OC[JCXG CP CTTC[NKMG VJKU QPG

```
B OQBSSBZ < >
```

+H [QW CNTGCF[JCXG /CVRNQVNKD KPUVCNNGF [QW ECP KORQTV KV YKVJ

```
JNQPSU NBUQMPUMJC QZQMPU BT QMU
*G ZPV SF VTJOH +VQZUFS /PUFCPPL ZPV NBZ BMTP XBOU UP SVO UIF GPMMP
MJOF PG DPEF UP EJTQMBZ ZPVS DPEF JO UIF OPUFCPPL
NBUQMPUMJC JOMJOF
```

#NN [QW PGGF VQ FQ VQ RNQV [QWT XCNWGU KU TWP

```
QMU QMPU B
*G ZPV BSF SVOOJOH GSPN B DPNNBOE MJOF ZPV NBZ OFFE UP EP UIJT
QMU TIPX
```

(QT GZCORNG [QW ECP RNQV C & CTTC[NKMG VJKU

```
Y   OQMJOTQBDF
Z   OQMJOTQBDF
QMUQMPU Y  ZQVSQMF MJOF
QMUQMPU Y  ZP          EPUT
```

9KVJ /CVRNQVNKD [QW JCXG CEEGUU VQ CP GPQTOQWU PWODGT QH XKUWC

```
GJH  QMUGJHVSF
BY  GJHBEE@TVCQMPU QSPKFDUJPO
9   OQBSBOHF
:   OQBSBOHF
9  :   OQNFTIHSJE 9  :
3   OQTRSU 9    :
;   OQTJO 3

BYQMPU@TVSGBDF 9  :  ;  STDUSJEF DNBQ WJSJEJT
```

6Q TGCF OQTG CDQWV /CVRNQVNKD CPF YJCV KV ECP FQ
KPUVCNNKPI /CVRNQVNKD UGG VJG QHHKEKCN FQEWOGPVCVKQP

*NBHF DSFEJUT +BZ "MBNNBS IUUQ KBMBNNBS HJUIVC JP

15.35.4 Future Changes

Shape-1 fields in dtypes won't be collapsed to scalars in a future version

Currently, a field specified as `[(name, dtype, 1)]` or `"1type"` is interpreted as a scalar field (i.e., the same as `[(name, dtype)]` or `[(name, dtype, ())]`). This now raises a FutureWarning; in a future version, it will be interpreted as a shape-(1,) field, i.e. the same as `[(name, dtype, (1,))]` or `"(1,)type"` (consistently with `[(name, dtype, n)]` / `"ntype"` with `n>1`, which is already equivalent to `[(name, dtype, (n,)]` / `"(n,)type"`).

15.35.5 Compatibility notes

`float16` subnormal rounding

Casting from a different floating point precision to `float16` used incorrect rounding in some edge cases. This means in rare cases, subnormal results will now be rounded up instead of down, changing the last bit (ULP) of the result.

Signed zero when using divmod

Starting in version *1.12.0*, numpy incorrectly returned a negatively signed zero when using the `divmod` and `floor_divide` functions when the result was zero. For example:

```
>>> np.zeros(10)//1
array([-0., -0., -0., -0., -0., -0., -0., -0., -0., -0.])
```

With this release, the result is correctly returned as a positively signed zero:

```
>>> np.zeros(10)//1
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

`MaskedArray.mask` now returns a view of the mask, not the mask itself

Returning the mask itself was unsafe, as it could be reshaped in place which would violate expectations of the masked array code. The behavior of `mask` is now consistent with `data`, which also returns a view.

The underlying mask can still be accessed with `._mask` if it is needed. Tests that contain `assert x.mask is not y.mask` or similar will need to be updated.

Do not lookup `__buffer__` attribute in `numpy.frombuffer`

Looking up `__buffer__` attribute in `numpy.frombuffer` was undocumented and non-functional. This code was removed. If needed, use `frombuffer(memoryview(obj), ...)` instead.

`out` is buffered for memory overlaps in `take`, `choose`, `put`

If the out argument to these functions is provided and has memory overlap with the other arguments, it is now buffered to avoid order-dependent behavior.

Unpickling while loading requires explicit opt-in

The functions `load`, and `lib.format.read_array` take an `allow_pickle` keyword which now defaults to `False` in response to CVE-2019-6446.

Potential changes to the random stream in old random module

Due to bugs in the application of `log` to random floating point numbers, the stream may change when sampling from `beta`, `binomial`, `laplace`, `logistic`, `logseries` or `multinomial` if a `0` is generated in the underlying `MT19937` random stream. There is a `1` in 10^{53} chance of this occurring, so the probability that the stream changes for any given seed is extremely small. If a `0` is encountered in the underlying generator, then the incorrect value produced (either `numpy.inf` or `numpy.nan`) is now dropped.

`i0` now always returns a result with the same shape as the input

Previously, the output was squeezed, such that, e.g., input with just a single element would lead to an array scalar being returned, and inputs with shapes such as `(10, 1)` would yield results that would not broadcast against the input.

Note that we generally recommend the SciPy implementation over the numpy one: it is a proper ufunc written in C, and more than an order of magnitude faster.

`can_cast` no longer assumes all unsafe casting is allowed

Previously, `can_cast` returned *True* for almost all inputs for `casting='unsafe'`, even for cases where casting was not possible, such as from a structured dtype to a regular one. This has been fixed, making it more consistent with actual casting using, e.g., the `.astype` method.

`ndarray.flags.writeable` can be switched to true slightly more often

In rare cases, it was not possible to switch an array from not writeable to writeable, although a base array is writeable. This can happen if an intermediate `ndarray.base` object is writeable. Previously, only the deepest base object was considered for this decision. However, in rare cases this object does not have the necessary information. In that case switching to writeable was never allowed. This has now been fixed.

15.35.6 C API changes

dimension or stride input arguments are now passed by `npy_intp const*`

Previously these function arguments were declared as the more strict `npy_intp*`, which prevented the caller passing constant data. This change is backwards compatible, but now allows code like:

```
npy_intp const fixed_dims[] = {1, 2, 3};
// no longer complains that the const-qualifier is discarded
npy_intp size = PyArray_MultiplyList(fixed_dims, 3);
```

15.35.7 New Features

New extensible `numpy.random` module with selectable random number generators

A new extensible `numpy.random` module along with four selectable random number generators and improved seeding designed for use in parallel processes has been added. The currently available *Bit Generators* are *MT19937*, *PCG64*, *Philox*, and *SFC64*. `PCG64` is the new default while `MT19937` is retained for backwards compatibility. Note that the legacy random module is unchanged and is now frozen, your current results will not change. More information is available in the API change description and in the `top-level view` documentation.

libFLAME

Support for building NumPy with the libFLAME linear algebra package as the LAPACK, implementation, see libFLAME for details.

User-defined BLAS detection order

`distutils` now uses an environment variable, comma-separated and case insensitive, to determine the detection order for BLAS libraries. By default `NPY_BLAS_ORDER=mkl,blis,openblas,atlas,accelerate,blas`. However, to force the use of OpenBLAS simply do:

```
NPY_BLAS_ORDER=openblas python setup.py build
```

which forces the use of OpenBLAS. This may be helpful for users which have a MKL installation but wishes to try out different implementations.

User-defined LAPACK detection order

`numpy.distutils` now uses an environment variable, comma-separated and case insensitive, to determine the detection order for LAPACK libraries. By default `NPY_LAPACK_ORDER=mkl,openblas,flame,atlas,accelerate,lapack`. However, to force the use of OpenBLAS simply do:

```
NPY_LAPACK_ORDER=openblas python setup.py build
```

which forces the use of OpenBLAS. This may be helpful for users which have a MKL installation but wishes to try out different implementations.

`ufunc.reduce` and related functions now accept a `where` mask

`ufunc.reduce`, `sum`, `prod`, `min`, `max` all now accept a `where` keyword argument, which can be used to tell which elements to include in the reduction. For reductions that do not have an identity, it is necessary to also pass in an initial value (e.g., `initial=np.inf` for `min`). For instance, the equivalent of `nansum` would be `np.sum(a, where=~np.isnan(a))`.

Timsort and radix sort have replaced mergesort for stable sorting

Both radix sort and timsort have been implemented and are now used in place of mergesort. Due to the need to maintain backward compatibility, the sorting `kind` options `"stable"` and `"mergesort"` have been made aliases of each other with the actual sort implementation depending on the array type. Radix sort is used for small integer types of 16 bits or less and timsort for the remaining types. Timsort features improved performance on data containing already or nearly sorted data and performs like mergesort on random data and requires $O(n/2)$ working space. Details of the timsort algorithm can be found at CPython listsort.txt.

`packbits` and `unpackbits` accept an `order` keyword

The `order` keyword defaults to `big`, and will order the **bits** accordingly. For `'order=big'` 3 will become `[0, 0, 0, 0, 0, 0, 1, 1]`, and `[1, 1, 0, 0, 0, 0, 0, 0]` for `order=little`

`unpackbits` now accepts a `count` parameter

`count` allows subsetting the number of bits that will be unpacked up-front, rather than reshaping and subsetting later, making the `packbits` operation invertible, and the unpacking less wasteful. Counts larger than the number of available bits add zero padding. Negative counts trim bits off the end instead of counting from the beginning. None counts implement the existing behavior of unpacking everything.

`linalg.svd` and `linalg.pinv` can be faster on hermitian inputs

These functions now accept a `hermitian` argument, matching the one added to `linalg.matrix_rank` in 1.14.0.

divmod operation is now supported for two `timedelta64` operands

The divmod operator now handles two `timedelta64` operands, with type signature `mm->qm`.

`fromfile` now takes an `offset` argument

This function now takes an `offset` keyword argument for binary files, which specifics the offset (in bytes) from the file's current position. Defaults to `0`.

New mode "empty" for `pad`

This mode pads an array to a desired shape without initializing the new entries.

`empty_like` and related functions now accept a `shape` argument

`empty_like`, `full_like`, `ones_like` and `zeros_like` now accept a `shape` keyword argument, which can be used to create a new array as the prototype, overriding its shape as well. This is particularly useful when combined with the `__array_function__` protocol, allowing the creation of new arbitrary-shape arrays from NumPy-like libraries when such an array is used as the prototype.

Floating point scalars implement `as_integer_ratio` to match the builtin float

This returns a (numerator, denominator) pair, which can be used to construct a `fractions.Fraction`.

Structured `dtype` objects can be indexed with multiple fields names

`arr.dtype[['a', 'b']]` now returns a dtype that is equivalent to `arr[['a', 'b']].dtype`, for consistency with `arr.dtype['a'] == arr['a'].dtype`.

Like the dtype of structured arrays indexed with a list of fields, this dtype has the same `itemsize` as the original, but only keeps a subset of the fields.

This means that `arr[['a', 'b']]` and `arr.view(arr.dtype[['a', 'b']])` are equivalent.

`.npy` files support unicode field names

A new format version of 3.0 has been introduced, which enables structured types with non-latin1 field names. This is used automatically when needed.

15.35.8 Improvements

Array comparison assertions include maximum differences

Error messages from array comparison tests such as `testing.assert_allclose` now include "max absolute difference" and "max relative difference," in addition to the previous "mismatch" percentage. This information makes it easier to update absolute and relative error tolerances.

Replacement of the fftpack based `fft` module by the pocketfft library

Both implementations have the same ancestor (Fortran77 FFTPACK by Paul N. Swarztrauber), but pocketfft contains additional modifications which improve both accuracy and performance in some circumstances. For FFT lengths containing large prime factors, pocketfft uses Bluestein's algorithm, which maintains $O(NlogN)$ run time complexity instead of deteriorating towards $O(N * N)$ for prime lengths. Also, accuracy for real valued FFTs with near prime lengths has improved and is on par with complex valued FFTs.

Further improvements to `ctypes` support in `numpy.ctypeslib`

A new `numpy.ctypeslib.as_ctypes_type` function has been added, which can be used to converts a `dtype` into a best-guess `ctypes` type. Thanks to this new function, `numpy.ctypeslib.as_ctypes` now supports a much wider range of array types, including structures, booleans, and integers of non-native endianness.

`numpy.errstate` is now also a function decorator

Currently, if you have a function like:

```
def foo():
    pass
```

and you want to wrap the whole thing in `errstate`, you have to rewrite it like so:

```
def foo():
    with np.errstate(...):
        pass
```

but with this change, you can do:

```
@np.errstate(...)
def foo():
    pass
```

thereby saving a level of indentation

`numpy.exp` and `numpy.log` speed up for float32 implementation

float32 implementation of `exp` and `log` now benefit from AVX2/AVX512 instruction set which are detected during runtime. `exp` has a max ulp error of 2.52 and `log` has a max ulp error or 3.83.

Improve performance of `numpy.pad`

The performance of the function has been improved for most cases by filling in a preallocated array with the desired padded shape instead of using concatenation.

`numpy.interp` handles infinities more robustly

In some cases where `interp` would previously return `nan`, it now returns an appropriate infinity.

Pathlib support for `fromfile`, *tofile* and `ndarray.dump`

`fromfile`, *ndarray.ndarray.tofile* and `ndarray.dump` now support the `pathlib.Path` type for the `file`/`fid` parameter.

Specialized `isnan`, `isinf`, and `isfinite` ufuncs for bool and int types

The boolean and integer types are incapable of storing `nan` and `inf` values, which allows us to provide specialized ufuncs that are up to 250x faster than the previous approach.

`isfinite` supports `datetime64` and `timedelta64` types

Previously, `isfinite` used to raise a *TypeError* on being used on these two types.

New keywords added to `nan_to_num`

`nan_to_num` now accepts keywords `nan`, `posinf` and `neginf` allowing the user to define the value to replace the `nan`, positive and negative `np.inf` values respectively.

MemoryErrors caused by allocated overly large arrays are more descriptive

Often the cause of a MemoryError is incorrect broadcasting, which results in a very large and incorrect shape. The message of the error now includes this shape to help diagnose the cause of failure.

`floor`, `ceil`, and `trunc` now respect builtin magic methods

These ufuncs now call the `__floor__`, `__ceil__`, and `__trunc__` methods when called on object arrays, making them compatible with `decimal.Decimal` and `fractions.Fraction` objects.

`quantile` now works on *fraction.Fraction* and `decimal.Decimal` objects

In general, this handles object arrays more gracefully, and avoids floating- point operations if exact arithmetic types are used.

Support of object arrays in `matmul`

It is now possible to use `matmul` (or the `@` operator) with object arrays. For instance, it is now possible to do:

```
from fractions import Fraction
a = np.array([[Fraction(1, 2), Fraction(1, 3)], [Fraction(1, 3), Fraction(1, 2)]])
b = a @ a
```

15.35.9 Changes

`median` and `percentile` family of functions no longer warn about `nan`

`numpy.median`, `numpy.percentile`, and `numpy.quantile` used to emit a `RuntimeWarning` when encountering an `nan`. Since they return the `nan` value, the warning is redundant and has been removed.

`timedelta64 % 0` behavior adjusted to return `NaT`

The modulus operation with two `np.timedelta64` operands now returns `NaT` in the case of division by zero, rather than returning zero

NumPy functions now always support overrides with `__array_function__`

NumPy now always checks the `__array_function__` method to implement overrides of NumPy functions on non-NumPy arrays, as described in NEP 18. The feature was available for testing with NumPy 1.16 if appropriate environment variables are set, but is now always enabled.

`lib.recfunctions.structured_to_unstructured` does not squeeze single-field views

Previously `structured_to_unstructured(arr[['a']])` would produce a squeezed result inconsistent with `structured_to_unstructured(arr[['a', b']])`. This was accidental. The old behavior can be retained with `structured_to_unstructured(arr[['a']]).squeeze(axis=-1)` or far more simply, `arr['a']`.

`clip` now uses a ufunc under the hood

This means that registering clip functions for custom dtypes in C via `descr->f->fastclip` is deprecated - they should use the ufunc registration mechanism instead, attaching to the `np.core.umath.clip` ufunc.

It also means that `clip` accepts `where` and `casting` arguments, and can be override with `__array_ufunc__`.

A consequence of this change is that some behaviors of the old `clip` have been deprecated:

- Passing `nan` to mean "do not clip" as one or both bounds. This didn't work in all cases anyway, and can be better handled by passing infinities of the appropriate sign.
- Using "unsafe" casting by default when an `out` argument is passed. Using `casting="unsafe"` explicitly will silence this warning.

Additionally, there are some corner cases with behavior changes:

- Padding `max < min` has changed to be more consistent across dtypes, but should not be relied upon.
- Scalar `min` and `max` take part in promotion rules like they do in all other ufuncs.

`__array_interface__` offset now works as documented

The interface may use an `offset` value that was mistakenly ignored.

Pickle protocol in `savez` set to 3 for `force zip64` flag

`savez` was not using the `force_zip64` flag, which limited the size of the archive to 2GB. But using the flag requires us to use pickle protocol 3 to write `object` arrays. The protocol used was bumped to 3, meaning the archive will be unreadable by Python2.

Structured arrays indexed with non-existent fields raise `KeyError` not `ValueError`

`arr['bad_field']` on a structured type raises `KeyError`, for consistency with `dict['bad_field']`.

15.36 NumPy 1.16.6 Release Notes

The NumPy 1.16.6 release fixes bugs reported against the 1.16.5 release, and also backports several enhancements from master that seem appropriate for a release series that is the last to support Python 2.7. The wheels on PyPI are linked with OpenBLAS v0.3.7, which should fix errors on Skylake series cpus.

Downstream developers building this release should use Cython >= 0.29.2 and, if using OpenBLAS, OpenBLAS >= v0.3.7. The supported Python versions are 2.7 and 3.5-3.7.

15.36.1 Highlights

- The `np.testing.utils` functions have been updated from 1.19.0-dev0. This improves the function documentation and error messages as well extending the `assert_array_compare` function to additional types.

15.36.2 New functions

Allow matmul (@ operator) to work with object arrays.

This is an enhancement that was added in NumPy 1.17 and seems reasonable to include in the LTS 1.16 release series.

15.36.3 Compatibility notes

Fix regression in matmul (@ operator) for boolean types

Booleans were being treated as integers rather than booleans, which was a regression from previous behavior.

15.36.4 Improvements

Array comparison assertions include maximum differences

Error messages from array comparison tests such as `testing.assert_allclose` now include "max absolute difference" and "max relative difference," in addition to the previous "mismatch" percentage. This information makes it easier to update absolute and relative error tolerances.

15.36.5 Contributors

A total of 10 people contributed to this release.

- CakeWithSteak
- Charles Harris
- Chris Burr
- Eric Wieser
- Fernando Saravia
- Lars Grueter
- Matti Picus
- Maxwell Aladago
- Qiming Sun
- Warren Weckesser

15.36.6 Pull requests merged

A total of 14 pull requests were merged for this release.

- #14211: BUG: Fix uint-overflow if padding with linear_ramp and negative…
- #14275: BUG: fixing to allow unpickling of PY3 pickles from PY2
- #14340: BUG: Fix misuse of .names and .fields in various places (backport…
- #14423: BUG: test, fix regression in converting to ctypes.
- #14434: BUG: Fixed maximum relative error reporting in assert_allclose
- #14509: BUG: Fix regression in boolean matmul.
- #14686: BUG: properly define PyArray_DescrCheck
- #14853: BLD: add 'apt update' to shippable
- #14854: BUG: Fix _ctypes class circular reference. (#13808)
- #14856: BUG: Fix *np.einsum* errors on Power9 Linux and z/Linux
- #14863: BLD: Prevent -flto from optimising long double representation…
- #14864: BUG: lib: Fix histogram problem with signed integer arrays.
- #15172: ENH: Backport improvements to testing functions.
- #15191: REL: Prepare for 1.16.6 release.

15.37 NumPy 1.16.5 Release Notes

The NumPy 1.16.5 release fixes bugs reported against the 1.16.4 release, and also backports several enhancements from master that seem appropriate for a release series that is the last to support Python 2.7. The wheels on PyPI are linked with OpenBLAS v0.3.7-dev, which should fix errors on Skylake series cpus.

Downstream developers building this release should use Cython >= 0.29.2 and, if using OpenBLAS, OpenBLAS >= v0.3.7. The supported Python versions are 2.7 and 3.5-3.7.

15.37.1 Contributors

A total of 18 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Alexander Shadchin
- Allan Haldane
- Bruce Merry +
- Charles Harris
- Colin Snyder +
- Dan Allan +
- Emile +
- Eric Wieser
- Grey Baker +
- Maksim Shabunin +
- Marten van Kerkwijk
- Matti Picus
- Peter Andreas Entschev +
- Ralf Gommers
- Richard Harris +
- Sebastian Berg
- Sergei Lebedev +
- Stephan Hoyer

15.37.2 Pull requests merged

A total of 23 pull requests were merged for this release.

- #13742: ENH: Add project URLs to setup.py
- #13823: TEST, ENH: fix tests and ctypes code for PyPy
- #13845: BUG: use npy_intp instead of int for indexing array
- #13867: TST: Ignore DeprecationWarning during nose imports
- #13905: BUG: Fix use-after-free in boolean indexing
- #13933: MAINT/BUG/DOC: Fix errors in _add_newdocs

- #13984: BUG: fix byte order reversal for datetime64[ns]
- #13994: MAINT,BUG: Use nbytes to also catch empty descr during allocation
- #14042: BUG: np.array cleared errors occurred in PyMemoryView_FromObject
- #14043: BUG: Fixes for Undefined Behavior Sanitizer (UBSan) errors.
- #14044: BUG: ensure that casting to/from structured is properly checked.
- #14045: MAINT: fix histogram*d dispatchers
- #14046: BUG: further fixup to histogram2d dispatcher.
- #14052: BUG: Replace contextlib.suppress for Python 2.7
- #14056: BUG: fix compilation of 3rd party modules with Py_LIMITED_API…
- #14057: BUG: Fix memory leak in dtype from dict constructor
- #14058: DOC: Document array_function at a higher level.
- #14084: BUG, DOC: add new recfunctions to *__all__*
- #14162: BUG: Remove stray print that causes a SystemError on python 3.7
- #14297: TST: Pin pytest version to 5.0.1.
- #14322: ENH: Enable huge pages in all Linux builds
- #14346: BUG: fix behavior of structured_to_unstructured on non-trivial…
- #14382: REL: Prepare for the NumPy 1.16.5 release.

15.38 NumPy 1.16.4 Release Notes

The NumPy 1.16.4 release fixes bugs reported against the 1.16.3 release, and also backports several enhancements from master that seem appropriate for a release series that is the last to support Python 2.7. The wheels on PyPI are linked with OpenBLAS v0.3.7-dev, which should fix issues on Skylake series cpus.

Downstream developers building this release should use Cython >= 0.29.2 and, if using OpenBLAS, OpenBLAS > v0.3.7. The supported Python versions are 2.7 and 3.5-3.7.

15.38.1 New deprecations

Writeable flag of C-API wrapped arrays

When an array is created from the C-API to wrap a pointer to data, the only indication we have of the read-write nature of the data is the `writeable` flag set during creation. It is dangerous to force the flag to writeable. In the future it will not be possible to switch the writeable flag to `True` from python. This deprecation should not affect many users since arrays created in such a manner are very rare in practice and only available through the NumPy C-API.

15.38.2 Compatibility notes

Potential changes to the random stream

Due to bugs in the application of log to random floating point numbers, the stream may change when sampling from `np.random.beta`, `np.random.binomial`, `np.random.laplace`, `np.random.logistic`, `np.random.logseries` or `np.random.multinomial` if a 0 is generated in the underlying MT19937 random stream. There is a 1 in 10^{53} chance of this occurring, and so the probability that the stream changes for any given seed is extremely small. If a 0 is encountered in the underlying generator, then the incorrect value produced (either `np.inf` or `np.nan`) is now dropped.

15.38.3 Changes

`numpy.lib.recfunctions.structured_to_unstructured` does not squeeze single-field views

Previously `structured_to_unstructured(arr[['a']])` would produce a squeezed result inconsistent with `structured_to_unstructured(arr[['a', b']])`. This was accidental. The old behavior can be retained with `structured_to_unstructured(arr[['a']]).squeeze(axis=-1)` or far more simply, `arr['a']`.

15.38.4 Contributors

A total of 10 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris
- Eric Wieser
- Dennis Zollo +
- Hunter Damron +
- Jingbei Li +
- Kevin Sheppard
- Matti Picus
- Nicola Soranzo +
- Sebastian Berg
- Tyler Reddy

15.38.5 Pull requests merged

A total of 16 pull requests were merged for this release.

- #13392: BUG: Some PyPy versions lack PyStructSequence_InitType2.
- #13394: MAINT, DEP: Fix deprecated `assertEquals()`
- #13396: BUG: Fix structured_to_unstructured on single-field types (backport)
- #13549: BLD: Make CI pass again with pytest 4.5
- #13552: TST: Register markers in conftest.py.
- #13559: BUG: Removes ValueError for empty kwargs in arraymultiter_new

- #13560: BUG: Add TypeError to accepted exceptions in crackfortran.
- #13561: BUG: Handle subarrays in descr_to_dtype
- #13562: BUG: Protect generators from log(0.0)
- #13563: BUG: Always return views from structured_to_unstructured when…
- #13564: BUG: Catch stderr when checking compiler version
- #13565: BUG: longdouble(int) does not work
- #13587: BUG: distutils/system_info.py fix missing subprocess import (#13523)
- #13620: BUG,DEP: Fix writeable flag setting for arrays without base
- #13641: MAINT: Prepare for the 1.16.4 release.
- #13644: BUG: special case object arrays when printing rel-, abs-error

15.39 NumPy 1.16.3 Release Notes

The NumPy 1.16.3 release fixes bugs reported against the 1.16.2 release, and also backports several enhancements from master that seem appropriate for a release series that is the last to support Python 2.7. The wheels on PyPI are linked with OpenBLAS v0.3.4+, which should fix the known threading issues found in previous OpenBLAS versions.

Downstream developers building this release should use Cython >= 0.29.2 and, if using OpenBLAS, OpenBLAS > v0.3.4.

The most noticeable change in this release is that unpickling object arrays when loading `*.npy` or `*.npz` files now requires an explicit opt-in. This backwards incompatible change was made in response to CVE-2019-6446.

15.39.1 Compatibility notes

Unpickling while loading requires explicit opt-in

The functions `np.load`, and `np.lib.format.read_array` take an *allow_pickle* keyword which now defaults to `False` in response to CVE-2019-6446.

15.39.2 Improvements

Covariance in *random.mvnormal* cast to double

This should make the tolerance used when checking the singular values of the covariance matrix more meaningful.

15.39.3 Changes

`__array_interface__` offset now works as documented

The interface may use an `offset` value that was previously mistakenly ignored.

15.40 NumPy 1.16.2 Release Notes

NumPy 1.16.2 is a quick release fixing several problems encountered on Windows. The Python versions supported are 2.7 and 3.5-3.7. The Windows problems addressed are:

- DLL load problems for NumPy wheels on Windows,
- distutils command line parsing on Windows.

There is also a regression fix correcting signed zeros produced by divmod, see below for details.

Downstream developers building this release should use Cython >= 0.29.2 and, if using OpenBLAS, OpenBLAS > v0.3.4.

If you are installing using pip, you may encounter a problem with older installed versions of NumPy that pip did not delete becoming mixed with the current version, resulting in an `ImportError`. That problem is particularly common on Debian derived distributions due to a modified pip. The fix is to make sure all previous NumPy versions installed by pip have been removed. See #12736 for discussion of the issue.

15.40.1 Compatibility notes

Signed zero when using divmod

Starting in version 1.12.0, numpy incorrectly returned a negatively signed zero when using the `divmod` and `floor_divide` functions when the result was zero. For example:

```
>>> np.zeros(10)//1
array([-0., -0., -0., -0., -0., -0., -0., -0., -0., -0.])
```

With this release, the result is correctly returned as a positively signed zero:

```
>>> np.zeros(10)//1
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

15.40.2 Contributors

A total of 5 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris
- Eric Wieser
- Matti Picus
- Tyler Reddy
- Tony LaTorre +

15.40.3 Pull requests merged

A total of 7 pull requests were merged for this release.

- #12909: TST: fix vmImage dispatch in Azure
- #12923: MAINT: remove complicated test of multiarray import failure mode
- #13020: BUG: fix signed zero behavior in npy_divmod
- #13026: MAINT: Add functions to parse shell-strings in the platform-native…
- #13028: BUG: Fix regression in parsing of F90 and F77 environment variables
- #13038: BUG: parse shell escaping in extra_compile_args and extra_link_args
- #13041: BLD: Windows absolute path DLL loading

15.41 NumPy 1.16.1 Release Notes

The NumPy 1.16.1 release fixes bugs reported against the 1.16.0 release, and also backports several enhancements from master that seem appropriate for a release series that is the last to support Python 2.7. The wheels on PyPI are linked with OpenBLAS v0.3.4+, which should fix the known threading issues found in previous OpenBLAS versions.

Downstream developers building this release should use Cython >= 0.29.2 and, if using OpenBLAS, OpenBLAS > v0.3.4.

If you are installing using pip, you may encounter a problem with older installed versions of NumPy that pip did not delete becoming mixed with the current version, resulting in an `ImportError`. That problem is particularly common on Debian derived distributions due to a modified pip. The fix is to make sure all previous NumPy versions installed by pip have been removed. See #12736 for discussion of the issue. Note that previously this problem resulted in an `AttributeError`.

15.41.1 Contributors

A total of 16 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Antoine Pitrou
- Arcesio Castaneda Medina +
- Charles Harris
- Chris Markiewicz +
- Christoph Gohlke
- Christopher J. Markiewicz +
- Daniel Hrisca +
- EelcoPeacs +
- Eric Wieser
- Kevin Sheppard
- Matti Picus
- OBATA Akio +
- Ralf Gommers
- Sebastian Berg

- Stephan Hoyer
- Tyler Reddy

15.41.2 Enhancements

- #12767: ENH: add mm->q floordiv
- #12768: ENH: port np.core.overrides to C for speed
- #12769: ENH: Add np.ctypeslib.as_ctypes_type(dtype), improve *np.ctypeslib.as_ctypes*
- #12773: ENH: add "max difference" messages to np.testing.assert_array_equal…
- #12820: ENH: Add mm->qm divmod
- #12890: ENH: add _dtype_ctype to namespace for freeze analysis

15.41.3 Compatibility notes

- The changed error message emitted by array comparison testing functions may affect doctests. See below for detail.
- Casting from double and single denormals to float16 has been corrected. In some rare cases, this may result in results being rounded up instead of down, changing the last bit (ULP) of the result.

15.41.4 New Features

divmod operation is now supported for two `timedelta64` operands

The divmod operator now handles two `np.timedelta64` operands, with type signature `mm->qm`.

15.41.5 Improvements

Further improvements to `ctypes` support in `np.ctypeslib`

A new `numpy.ctypeslib.as_ctypes_type` function has been added, which can be used to converts a *dtype* into a best-guess `ctypes` type. Thanks to this new function, `numpy.ctypeslib.as_ctypes` now supports a much wider range of array types, including structures, booleans, and integers of non-native endianness.

Array comparison assertions include maximum differences

Error messages from array comparison tests such as *np.testing.assert_allclose* now include "max absolute difference" and "max relative difference," in addition to the previous "mismatch" percentage. This information makes it easier to update absolute and relative error tolerances.

15.41.6 Changes

`timedelta64 % 0` behavior adjusted to return `NaT`

The modulus operation with two `np.timedelta64` operands now returns `NaT` in the case of division by zero, rather than returning zero

15.42 NumPy 1.16.0 Release Notes

This NumPy release is the last one to support Python 2.7 and will be maintained as a long term release with bug fixes until 2020. Support for Python 3.4 been dropped, the supported Python versions are 2.7 and 3.5-3.7. The wheels on PyPI are linked with OpenBLAS v0.3.4+, which should fix the known threading issues found in previous OpenBLAS versions.

Downstream developers building this release should use Cython >= 0.29 and, if using OpenBLAS, OpenBLAS > v0.3.4.

This release has seen a lot of refactoring and features many bug fixes, improved code organization, and better cross platform compatibility. Not all of these improvements will be visible to users, but they should help make maintenance easier going forward.

15.42.1 Highlights

- Experimental (opt-in only) support for overriding numpy functions, see `__array_function__` below.
- The `matmul` function is now a ufunc. This provides better performance and allows overriding with `__array_ufunc__`.
- Improved support for the ARM and POWER architectures.
- Improved support for AIX and PyPy.
- Improved interop with ctypes.
- Improved support for PEP 3118.

15.42.2 New functions

- New functions added to the *numpy.lib.recfuntions* module to ease the structured assignment changes:
  - `assign_fields_by_name`
  - `structured_to_unstructured`
  - `unstructured_to_structured`
  - `apply_along_fields`
  - `require_fields`

  See the user guide at <https://docs.scipy.org/doc/numpy/user/basics.rec.html> for more info.

15.42.3 New deprecations

- The type dictionaries *numpy.core.typeNA* and *numpy.core.sctypeNA* are deprecated. They were buggy and not documented and will be removed in the 1.18 release. Use'numpy.sctypeDict' instead.
- The *numpy.asscalar* function is deprecated. It is an alias to the more powerful `numpy.ndarray.item`, not tested, and fails for scalars.
- The *numpy.set_array_ops* and *numpy.get_array_ops* functions are deprecated. As part of *NEP 15*, they have been deprecated along with the C-API functions `PyArray_SetNumericOps` and `PyArray_GetNumericOps`. Users who wish to override the inner loop functions in built-in ufuncs should use `PyUFunc_ReplaceLoopBySignature`.
- The `numpy.unravel_index` keyword argument `dims` is deprecated, use `shape` instead.
- The `numpy.histogram` `normed` argument is deprecated. It was deprecated previously, but no warning was issued.
- The `positive` operator (+) applied to non-numerical arrays is deprecated. See below for details.
- Passing an iterator to the stack functions is deprecated

15.42.4 Expired deprecations

- NaT comparisons now return `False` without a warning, finishing a deprecation cycle begun in NumPy 1.11.
- `np.lib.function_base.unique` was removed, finishing a deprecation cycle begun in NumPy 1.4. Use `numpy.unique` instead.
- multi-field indexing now returns views instead of copies, finishing a deprecation cycle begun in NumPy 1.7. The change was previously attempted in NumPy 1.14 but reverted until now.
- `np.PackageLoader` and `np.pkgload` have been removed. These were deprecated in 1.10, had no tests, and seem to no longer work in 1.15.

15.42.5 Future changes

- NumPy 1.17 will drop support for Python 2.7.

15.42.6 Compatibility notes

f2py script on Windows

On Windows, the installed script for running f2py is now an `.exe` file rather than a `*.py` file and should be run from the command line as `f2py` whenever the `Scripts` directory is in the path. Running `f2py` as a module `python -m numpy.f2py [...]` will work without path modification in any version of NumPy.

NaT comparisons

Consistent with the behavior of NaN, all comparisons other than inequality checks with datetime64 or timedelta64 NaT ("not-a-time") values now always return `False`, and inequality checks with NaT now always return `True`. This includes comparisons between NaT values. For compatibility with the old behavior, use `np.isnat` to explicitly check for NaT or convert datetime64/timedelta64 arrays with `.astype(np.int64)` before making comparisons.

complex64/128 alignment has changed

The memory alignment of complex types is now the same as a C-struct composed of two floating point values, while before it was equal to the size of the type. For many users (for instance on x64/unix/gcc) this means that complex64 is now 4-byte aligned instead of 8-byte aligned. An important consequence is that aligned structured dtypes may now have a different size. For instance, `np.dtype('c8,u1', align=True)` used to have an itemsize of 16 (on x64/gcc) but now it is 12.

More in detail, the complex64 type now has the same alignment as a C-struct `struct {float r, i;}`, according to the compiler used to compile numpy, and similarly for the complex128 and complex256 types.

nd_grid __len__ removal

`len(np.mgrid)` and `len(np.ogrid)` are now considered nonsensical and raise a `TypeError`.

`np.unravel_index` now accepts `shape` keyword argument

Previously, only the `dims` keyword argument was accepted for specification of the shape of the array to be used for unraveling. `dims` remains supported, but is now deprecated.

multi-field views return a view instead of a copy

Indexing a structured array with multiple fields, e.g., `arr[['f1', 'f3']]`, returns a view into the original array instead of a copy. The returned view will often have extra padding bytes corresponding to intervening fields in the original array, unlike before, which will affect code such as `arr[['f1', 'f3']].view('float64')`. This change has been planned since numpy 1.7. Operations hitting this path have emitted `FutureWarnings` since then. Additional `FutureWarnings` about this change were added in 1.12.

To help users update their code to account for these changes, a number of functions have been added to the `numpy.lib.recfunctions` module which safely allow such operations. For instance, the code above can be replaced with `structured_to_unstructured(arr[['f1', 'f3']], dtype='float64')`. See the "accessing multiple fields" section of the user guide.

15.42.7 C API changes

The `NPY_FEATURE_VERSION` was incremented to 0x0000D, due to the addition of:

- `PyUFuncObject.core_dim_flags`
- `PyUFuncObject.core_dim_sizes`
- `PyUFuncObject.identity_value`
- `PyUFunc_FromFuncAndDataAndSignatureAndIdentity`

15.42.8 New Features

Integrated squared error (ISE) estimator added to `histogram`

This method (`bins='stone'`) for optimizing the bin number is a generalization of the Scott's rule. The Scott's rule assumes the distribution is approximately Normal, while the ISE is a non-parametric method based on cross-validation.

`max_rows` keyword added for `np.loadtxt`

New keyword `max_rows` in `numpy.loadtxt` sets the maximum rows of the content to be read after `skiprows`, as in `numpy.genfromtxt`.

modulus operator support added for `np.timedelta64` operands

The modulus (remainder) operator is now supported for two operands of type `np.timedelta64`. The operands may have different units and the return value will match the type of the operands.

15.42.9 Improvements

no-copy pickling of numpy arrays

Up to protocol 4, numpy array pickling created 2 spurious copies of the data being serialized. With pickle protocol 5, and the `PickleBuffer` API, a large variety of numpy arrays can now be serialized without any copy using out-of-band buffers, and with one less copy using in-band buffers. This results, for large arrays, in an up to 66% drop in peak memory usage.

build shell independence

NumPy builds should no longer interact with the host machine shell directly. `exec_command` has been replaced with `subprocess.check_output` where appropriate.

np.polynomial.Polynomial classes render in LaTeX in Jupyter notebooks

When used in a front-end that supports it, *Polynomial* instances are now rendered through LaTeX. The current format is experimental, and is subject to change.

`randint` and `choice` now work on empty distributions

Even when no elements needed to be drawn, `np.random.randint` and `np.random.choice` raised an error when the arguments described an empty distribution. This has been fixed so that e.g. `np.random.choice([], 0) == np.array([], dtype=float64)`.

`linalg.lstsq`, `linalg.qr`, and `linalg.svd` now work with empty arrays

Previously, a `LinAlgError` would be raised when an empty matrix/empty matrices (with zero rows and/or columns) is/are passed in. Now outputs of appropriate shapes are returned.

Chain exceptions to give better error messages for invalid PEP3118 format strings

This should help track down problems.

Einsum optimization path updates and efficiency improvements

Einsum was synchronized with the current upstream work.

`numpy.angle` and `numpy.expand_dims` now work on `ndarray` subclasses

In particular, they now work for masked arrays.

`NPY_NO_DEPRECATED_API` compiler warning suppression

Setting `NPY_NO_DEPRECATED_API` to a value of 0 will suppress the current compiler warnings when the deprecated numpy API is used.

`np.diff` Added kwargs prepend and append

New kwargs `prepend` and `append`, allow for values to be inserted on either end of the differences. Similar to options for *ediff1d*. Now the inverse of *cumsum* can be obtained easily via `prepend=0`.

ARM support updated

Support for ARM CPUs has been updated to accommodate 32 and 64 bit targets, and also big and little endian byte ordering. AARCH32 memory alignment issues have been addressed. CI testing has been expanded to include AARCH64 targets via the services of shippable.com.

Appending to build flags

`numpy.distutils` has always overridden rather than appended to *LDFLAGS* and other similar such environment variables for compiling Fortran extensions. Now, if the *NPY_DISTUTILS_APPEND_FLAGS* environment variable is set to 1, the behavior will be appending. This applied to: *LDFLAGS*, *F77FLAGS*, *F90FLAGS*, *FREEFLAGS*, *FOPT*, *FDEBUG*, and *FFLAGS*. See gh-11525 for more details.

Generalized ufunc signatures now allow fixed-size dimensions

By using a numerical value in the signature of a generalized ufunc, one can indicate that the given function requires input or output to have dimensions with the given size. E.g., the signature of a function that converts a polar angle to a two-dimensional cartesian unit vector would be `()->(2)`; that for one that converts two spherical angles to a three-dimensional unit vector would be `(),()->(3)`; and that for the cross product of two three-dimensional vectors would be `(3),(3)->(3)`.

Note that to the elementary function these dimensions are not treated any differently from variable ones indicated with a name starting with a letter; the loop still is passed the corresponding size, but it can now count on that size being equal to the fixed one given in the signature.

Generalized ufunc signatures now allow flexible dimensions

Some functions, in particular numpy's implementation of `@` as `matmul`, are very similar to generalized ufuncs in that they operate over core dimensions, but one could not present them as such because they were able to deal with inputs in which a dimension is missing. To support this, it is now allowed to postfix a dimension name with a question mark to indicate that the dimension does not necessarily have to be present.

With this addition, the signature for `matmul` can be expressed as `(m?,n),(n,p?)->(m?,p?)`. This indicates that if, e.g., the second operand has only one dimension, for the purposes of the elementary function it will be treated as if that input has core shape `(n, 1)`, and the output has the corresponding core shape of `(m, 1)`. The actual output array, however, has the flexible dimension removed, i.e., it will have shape `(..., m)`. Similarly, if both arguments have only a single dimension, the inputs will be presented as having shapes `(1, n)` and `(n, 1)` to the elementary function, and the output as `(1, 1)`, while the actual output array returned will have shape `()`. In this way, the signature allows one to use a single elementary function for four related but different signatures, `(m,n),(n,p)->(m,p)`, `(n),(n,p)->(p)`, `(m,n),(n)->(m)` and `(n),(n)->()`.

`np.clip` and the `clip` method check for memory overlap

The `out` argument to these functions is now always tested for memory overlap to avoid corrupted results when memory overlap occurs.

New value `unscaled` for option `cov` in `np.polyfit`

A further possible value has been added to the `cov` parameter of the `np.polyfit` function. With `cov='unscaled'` the scaling of the covariance matrix is disabled completely (similar to setting `absolute_sigma=True` in `scipy.optimize.curve_fit`). This would be useful in occasions, where the weights are given by 1/sigma with sigma being the (known) standard errors of (Gaussian distributed) data points, in which case the unscaled matrix is already a correct estimate for the covariance matrix.

Detailed docstrings for scalar numeric types

The `help` function, when applied to numeric types such as `numpy.intc`, `numpy.int_`, and `numpy.longlong`, now lists all of the aliased names for that type, distinguishing between platform -dependent and -independent aliases.

`__module__` attribute now points to public modules

The `__module__` attribute on most NumPy functions has been updated to refer to the preferred public module from which to access a function, rather than the module in which the function happens to be defined. This produces more informative displays for functions in tools such as IPython, e.g., instead of `<function 'numpy.core.fromnumeric.sum'>` you now see `<function 'numpy.sum'>`.

Large allocations marked as suitable for transparent hugepages

On systems that support transparent hugepages over the madvise system call numpy now marks that large memory allocations can be backed by hugepages which reduces page fault overhead and can in some fault heavy cases improve performance significantly. On Linux the setting for huge pages to be used, */sys/kernel/mm/transparent_hugepage/enabled*, must be at least *madvise*. Systems which already have it set to *always* will not see much difference as the kernel will automatically use huge pages where appropriate.

Users of very old Linux kernels (~3.x and older) should make sure that */sys/kernel/mm/transparent_hugepage/defrag* is not set to *always* to avoid performance problems due concurrency issues in the memory defragmentation.

Alpine Linux (and other musl c library distros) support

We now default to use *fenv.h* for floating point status error reporting. Previously we had a broken default that sometimes would not report underflow, overflow, and invalid floating point operations. Now we can support non-glibc distributions like Alpine Linux as long as they ship *fenv.h*.

Speedup `np.block` for large arrays

Large arrays (greater than `512 * 512`) now use a blocking algorithm based on copying the data directly into the appropriate slice of the resulting array. This results in significant speedups for these large arrays, particularly for arrays being blocked along more than 2 dimensions.

`arr.ctypes.data_as(...)` holds a reference to arr

Previously the caller was responsible for keeping the array alive for the lifetime of the pointer.

Speedup `np.take` for read-only arrays

The implementation of `np.take` no longer makes an unnecessary copy of the source array when its `writeable` flag is set to `False`.

Support path-like objects for more functions

The `np.core.records.fromfile` function now supports `pathlib.Path` and other path-like objects in addition to a file object. Furthermore, the `np.load` function now also supports path-like objects when using memory mapping (`mmap_mode` keyword argument).

Better behaviour of ufunc identities during reductions

Universal functions have an `.identity` which is used when `.reduce` is called on an empty axis.

As of this release, the logical binary ufuncs, *logical_and*, *logical_or*, and *logical_xor*, now have `identity` s of type *bool*, where previously they were of type *int*. This restores the 1.14 behavior of getting `bool` s when reducing empty object arrays with these ufuncs, while also keeping the 1.15 behavior of getting `int` s when reducing empty object arrays with arithmetic ufuncs like `add` and `multiply`.

Additionally, *logaddexp* now has an identity of `-inf`, allowing it to be called on empty sequences, where previously it could not be.

This is possible thanks to the new `PyUFunc_FromFuncAndDataAndSignatureAndIdentity`, which allows arbitrary values to be used as identities now.

Improved conversion from ctypes objects

Numpy has always supported taking a value or type from `ctypes` and converting it into an array or dtype, but only behaved correctly for simpler types. As of this release, this caveat is lifted - now:

- The `_pack_` attribute of `ctypes.Structure`, used to emulate C's `__attribute__((packed))`, is respected.
- Endianness of all ctypes objects is preserved
- `ctypes.Union` is supported
- Non-representable constructs raise exceptions, rather than producing dangerously incorrect results:
  - Bitfields are no longer interpreted as sub-arrays
  - Pointers are no longer replaced with the type that they point to

A new `ndpointer.contents` member

This matches the `.contents` member of normal ctypes arrays, and can be used to construct an `np.array` around the pointers contents. This replaces `np.array(some_nd_pointer)`, which stopped working in 1.15. As a side effect of this change, `ndpointer` now supports dtypes with overlapping fields and padding.

`matmul` is now a `ufunc`

`numpy.matmul` is now a ufunc which means that both the function and the `__matmul__` operator can now be overridden by `__array_ufunc__`. Its implementation has also changed. It uses the same BLAS routines as `numpy.dot`, ensuring its performance is similar for large matrices.

Start and stop arrays for `linspace`, `logspace` and `geomspace`

These functions used to be limited to scalar stop and start values, but can now take arrays, which will be properly broadcast and result in an output which has one axis prepended. This can be used, e.g., to obtain linearly interpolated points between sets of points.

CI extended with additional services

We now use additional free CI services, thanks to the companies that provide:

- Codecoverage testing via codecov.io
- Arm testing via shippable.com
- Additional test runs on azure pipelines

These are in addition to our continued use of travis, appveyor (for wheels) and LGTM

15.42.10 Changes

Comparison ufuncs will now error rather than return NotImplemented

Previously, comparison ufuncs such as `np.equal` would return *NotImplemented* if their arguments had structured dtypes, to help comparison operators such as `__eq__` deal with those. This is no longer needed, as the relevant logic has moved to the comparison operators proper (which thus do continue to return *NotImplemented* as needed). Hence, like all other ufuncs, the comparison ufuncs will now error on structured dtypes.

Positive will now raise a deprecation warning for non-numerical arrays

Previously, `+array` unconditionally returned a copy. Now, it will raise a `DeprecationWarning` if the array is not numerical (i.e., if `np.positive(array)` raises a `TypeError`. For `ndarray` subclasses that override the default `__array_ufunc__` implementation, the `TypeError` is passed on.

`NDArrayOperatorsMixin` now implements matrix multiplication

Previously, `np.lib.mixins.NDArrayOperatorsMixin` did not implement the special methods for Python's matrix multiplication operator (`@`). This has changed now that `matmul` is a ufunc and can be overridden using `__array_ufunc__`.

The scaling of the covariance matrix in `np.polyfit` is different

So far, `np.polyfit` used a non-standard factor in the scaling of the the covariance matrix. Namely, rather than using the standard `chisq/(M-N)`, it scaled it with `chisq/(M-N-2)` where M is the number of data points and N is the number of parameters. This scaling is inconsistent with other fitting programs such as e.g. `scipy.optimize.curve_fit` and was changed to `chisq/(M-N)`.

`maximum` and `minimum` no longer emit warnings

As part of code introduced in 1.10, `float32` and `float64` set invalid float status when a Nan is encountered in `numpy.maximum` and `numpy.minimum`, when using SSE2 semantics. This caused a *RuntimeWarning* to sometimes be emitted. In 1.15 we fixed the inconsistencies which caused the warnings to become more conspicuous. Now no warnings will be emitted.

Umath and multiarray c-extension modules merged into a single module

The two modules were merged, according to NEP 15. Previously *np.core.umath* and *np.core.multiarray* were separate c-extension modules. They are now python wrappers to the single *np.core/_multiarray_math* c-extension module.

`getfield` validity checks extended

`numpy.ndarray.getfield` now checks the dtype and offset arguments to prevent accessing invalid memory locations.

NumPy functions now support overrides with `__array_function__`

NumPy has a new experimental mechanism for overriding the implementation of almost all NumPy functions on non-NumPy arrays by defining an `__array_function__` method, as described in NEP 18.

This feature is not yet been enabled by default, but has been released to facilitate experimentation by potential users. See the NEP for details on setting the appropriate environment variable. We expect the NumPy 1.17 release will enable overrides by default, which will also be more performant due to a new implementation written in C.

Arrays based off readonly buffers cannot be set `writeable`

We now disallow setting the `writeable` flag True on arrays created from `fromstring(readonly-buffer)`.

15.43 NumPy 1.15.4 Release Notes

This is a bugfix release for bugs and regressions reported following the 1.15.3 release. The Python versions supported by this release are 2.7, 3.4-3.7. The wheels are linked with OpenBLAS v0.3.0, which should fix some of the linalg problems reported for NumPy 1.14.

15.43.1 Compatibility Note

The NumPy 1.15.x OS X wheels released on PyPI no longer contain 32-bit binaries. That will also be the case in future releases. See #11625 for the related discussion. Those needing 32-bit support should look elsewhere or build from source.

15.43.2 Contributors

A total of 4 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris
- Matti Picus
- Sebastian Berg
- bbbbbbbbba +

15.43.3 Pull requests merged

A total of 4 pull requests were merged for this release.

- #12296: BUG: Dealloc cached buffer info
- #12297: BUG: Fix fill value in masked array '==' and '!=' ops.
- #12307: DOC: Correct the default value of *optimize* in `numpy.einsum`
- #12320: REL: Prepare for the NumPy 1.15.4 release

15.44 NumPy 1.15.3 Release Notes

This is a bugfix release for bugs and regressions reported following the 1.15.2 release. The Python versions supported by this release are 2.7, 3.4-3.7. The wheels are linked with OpenBLAS v0.3.0, which should fix some of the linalg problems reported for NumPy 1.14.

15.44.1 Compatibility Note

The NumPy 1.15.x OS X wheels released on PyPI no longer contain 32-bit binaries. That will also be the case in future releases. See #11625 for the related discussion. Those needing 32-bit support should look elsewhere or build from source.

15.44.2 Contributors

A total of 7 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Allan Haldane
- Charles Harris
- Jeroen Demeyer
- Kevin Sheppard
- Matthew Bowden +
- Matti Picus
- Tyler Reddy

15.44.3 Pull requests merged

A total of 12 pull requests were merged for this release.

- #12080: MAINT: Blacklist some MSVC complex functions.
- #12083: TST: Add azure CI testing to 1.15.x branch.
- #12084: BUG: test_path() now uses Path.resolve()
- #12085: TST, MAINT: Fix some failing tests on azure-pipelines mac and…
- #12187: BUG: Fix memory leak in mapping.c
- #12188: BUG: Allow boolean subtract in histogram
- #12189: BUG: Fix in-place permutation

- #12190: BUG: limit default for get_num_build_jobs() to 8
- #12191: BUG: OBJECT_to_* should check for errors
- #12192: DOC: Prepare for NumPy 1.15.3 release.
- #12237: BUG: Fix MaskedArray fill_value type conversion.
- #12238: TST: Backport azure-pipeline testing fixes for Mac

15.45 NumPy 1.15.2 Release Notes

This is a bugfix release for bugs and regressions reported following the 1.15.1 release.

- The matrix PendingDeprecationWarning is now suppressed in pytest 3.8.
- The new cached allocations machinery has been fixed to be thread safe.
- The boolean indexing of subclasses now works correctly.
- A small memory leak in PyArray_AdaptFlexibleDType has been fixed.

The Python versions supported by this release are 2.7, 3.4-3.7. The wheels are linked with OpenBLAS v0.3.0, which should fix some of the linalg problems reported for NumPy 1.14.

15.45.1 Compatibility Note

The NumPy 1.15.x OS X wheels released on PyPI no longer contain 32-bit binaries. That will also be the case in future releases. See #11625 for the related discussion. Those needing 32-bit support should look elsewhere or build from source.

15.45.2 Contributors

A total of 4 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris
- Julian Taylor
- Marten van Kerkwijk
- Matti Picus

15.45.3 Pull requests merged

A total of 4 pull requests were merged for this release.

- #11902: BUG: Fix matrix PendingDeprecationWarning suppression for pytest…
- #11981: BUG: fix cached allocations without the GIL for 1.15.x
- #11982: BUG: fix refcount leak in PyArray_AdaptFlexibleDType
- #11992: BUG: Ensure boolean indexing of subclasses sets base correctly.

15.46 NumPy 1.15.1 Release Notes

This is a bugfix release for bugs and regressions reported following the 1.15.0 release.

- The annoying but harmless RuntimeWarning that "numpy.dtype size changed" has been suppressed. The long standing suppression was lost in the transition to pytest.
- The update to Cython 0.28.3 exposed a problematic use of a gcc attribute used to prefer code size over speed in module initialization, possibly resulting in incorrect compiled code. This has been fixed in latest Cython but has been disabled here for safety.
- Support for big-endian and ARMv8 architectures has been improved.

The Python versions supported by this release are 2.7, 3.4-3.7. The wheels are linked with OpenBLAS v0.3.0, which should fix some of the linalg problems reported for NumPy 1.14.

15.46.1 Compatibility Note

The NumPy 1.15.x OS X wheels released on PyPI no longer contain 32-bit binaries. That will also be the case in future releases. See #11625 for the related discussion. Those needing 32-bit support should look elsewhere or build from source.

15.46.2 Contributors

A total of 7 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris
- Chris Billington
- Elliott Sales de Andrade +
- Eric Wieser
- Jeremy Manning +
- Matti Picus
- Ralf Gommers

15.46.3 Pull requests merged

A total of 24 pull requests were merged for this release.

- #11647: MAINT: Filter Cython warnings in `__init__.py`
- #11648: BUG: Fix doc source links to unwrap decorators
- #11657: BUG: Ensure singleton dimensions are not dropped when converting…
- #11661: BUG: Warn on Nan in minimum,maximum for scalars
- #11665: BUG: cython sometimes emits invalid gcc attribute
- #11682: BUG: Fix regression in void_getitem
- #11698: BUG: Make matrix_power again work for object arrays.
- #11700: BUG: Add missing PyErr_NoMemory after failing malloc
- #11719: BUG: Fix undefined functions on big-endian systems.

- #11720: MAINT: Make einsum optimize default to False.
- #11746: BUG: Fix regression in loadtxt for bz2 text files in Python 2.
- #11757: BUG: Revert use of *console_scripts*.
- #11758: BUG: Fix Fortran kind detection for aarch64 & s390x.
- #11759: BUG: Fix printing of longdouble on ppc64le.
- #11760: BUG: Fixes for unicode field names in Python 2
- #11761: BUG: Increase required cython version on python 3.7
- #11763: BUG: check return value of _buffer_format_string
- #11775: MAINT: Make assert_array_compare more generic.
- #11776: TST: Fix urlopen stubbing.
- #11777: BUG: Fix regression in intersect1d.
- #11779: BUG: Fix test sensitive to platform byte order.
- #11781: BUG: Avoid signed overflow in histogram
- #11785: BUG: Fix pickle and memoryview for datetime64, timedelta64 scalars
- #11786: BUG: Deprecation triggers segfault

15.47 NumPy 1.15.0 Release Notes

NumPy 1.15.0 is a release with an unusual number of cleanups, many deprecations of old functions, and improvements to many existing functions. Please read the detailed descriptions below to see if you are affected.

For testing, we have switched to pytest as a replacement for the no longer maintained nose framework. The old nose based interface remains for downstream projects who may still be using it.

The Python versions supported by this release are 2.7, 3.4-3.7. The wheels are linked with OpenBLAS v0.3.0, which should fix some of the linalg problems reported for NumPy 1.14.

15.47.1 Highlights

- NumPy has switched to pytest for testing.
- A new `numpy.printoptions` context manager.
- Many improvements to the histogram functions.
- Support for unicode field names in python 2.7.
- Improved support for PyPy.
- Fixes and improvements to `numpy.einsum`.

15.47.2 New functions

- `numpy.gcd` and `numpy.lcm`, to compute the greatest common divisor and least common multiple.
- `numpy.ma.stack`, the `numpy.stack` array-joining function generalized to masked arrays.
- `numpy.quantile` function, an interface to `percentile` without factors of 100
- `numpy.nanquantile` function, an interface to `nanpercentile` without factors of 100
- `numpy.printoptions`, a context manager that sets print options temporarily for the scope of the `with` block:

```
>>> with np.printoptions(precision=2):
...     print(np.array([2.0]) / 3)
[0.67]
```

- `numpy.histogram_bin_edges`, a function to get the edges of the bins used by a histogram without needing to calculate the histogram.
- C functions *npy_get_floatstatus_barrier* and *npy_clear_floatstatus_barrier* have been added to deal with compiler optimization changing the order of operations. See below for details.

15.47.3 Deprecations

- Aliases of builtin `pickle` functions are deprecated, in favor of their unaliased `pickle.<func>` names:
  - *numpy.loads*
  - *numpy.core.numeric.load*
  - *numpy.core.numeric.loads*
  - *numpy.ma.loads*, *numpy.ma.dumps*
  - *numpy.ma.load*, *numpy.ma.dump* - these functions already failed on python 3 when called with a string.
- Multidimensional indexing with anything but a tuple is deprecated. This means that the index list in `ind = [slice(None), 0]; arr[ind]` should be changed to a tuple, e.g., `ind = [slice(None), 0]; arr[tuple(ind)]` or `arr[(slice(None), 0)]`. That change is necessary to avoid ambiguity in expressions such as `arr[[[0, 1], [0, 1]]]`, currently interpreted as `arr[array([0, 1]), array([0, 1])]`, that will be interpreted as `arr[array([[0, 1], [0, 1]])]` in the future.
- Imports from the following sub-modules are deprecated, they will be removed at some future date.
  - `numpy.testing.utils`
  - *numpy.testing.decorators*
  - *numpy.testing.nosetester*
  - *numpy.testing.noseclasses*
  - `numpy.core.umath_tests`
- Giving a generator to `numpy.sum` is now deprecated. This was undocumented behavior, but worked. Previously, it would calculate the sum of the generator expression. In the future, it might return a different result. Use `np.sum(np.from_iter(generator))` or the built-in Python `sum` instead.
- Users of the C-API should call `PyArrayResolveWriteBackIfCopy` or `PyArray_DiscardWritbackIfCopy` on any array with the `WRITEBACKIFCOPY` flag set, before deallocating the array. A deprecation warning will be emitted if those calls are not used when needed.

- Users of `nditer` should use the nditer object as a context manager anytime one of the iterator operands is writeable, so that numpy can manage writeback semantics, or should call `it.close()`. A *RuntimeWarning* may be emitted otherwise in these cases.
- The `normed` argument of `np.histogram`, deprecated long ago in 1.6.0, now emits a `DeprecationWarning`.

15.47.4 Future Changes

- NumPy 1.16 will drop support for Python 3.4.
- NumPy 1.17 will drop support for Python 2.7.

15.47.5 Compatibility notes

Compiled testing modules renamed and made private

The following compiled modules have been renamed and made private:

- `umath_tests` -> `_umath_tests`
- `test_rational` -> `_rational_tests`
- `multiarray_tests` -> `_multiarray_tests`
- `struct_ufunc_test` -> `_struct_ufunc_tests`
- `operand_flag_tests` -> `_operand_flag_tests`

The `umath_tests` module is still available for backwards compatibility, but will be removed in the future.

The `NpzFile` returned by `np.savez` is now a `collections.abc.Mapping`

This means it behaves like a readonly dictionary, and has a new `.values()` method and `len()` implementation.

For python 3, this means that `.iteritems()`, `.iterkeys()` have been deprecated, and `.keys()` and `.items()` now return views and not lists. This is consistent with how the builtin `dict` type changed between python 2 and python 3.

Under certain conditions, `nditer` must be used in a context manager

When using an `numpy.nditer` with the `"writeonly"` or `"readwrite"` flags, there are some circumstances where nditer doesn't actually give you a view of the writable array. Instead, it gives you a copy, and if you make changes to the copy, nditer later writes those changes back into your actual array. Currently, this writeback occurs when the array objects are garbage collected, which makes this API error-prone on CPython and entirely broken on PyPy. Therefore, `nditer` should now be used as a context manager whenever it is used with writeable arrays, e.g., `with np.nditer(...) as it: ...`. You may also explicitly call `it.close()` for cases where a context manager is unusable, for instance in generator expressions.

Numpy has switched to using pytest instead of nose for testing

The last nose release was 1.3.7 in June, 2015, and development of that tool has ended, consequently NumPy has now switched to using pytest. The old decorators and nose tools that were previously used by some downstream projects remain available, but will not be maintained. The standard testing utilities, `assert_almost_equal` and such, are not be affected by this change except for the nose specific functions `import_nose` and `raises`. Those functions are not used in numpy, but are kept for downstream compatibility.

Numpy no longer monkey-patches `ctypes` with `__array_interface__`

Previously numpy added `__array_interface__` attributes to all the integer types from `ctypes`.

`np.ma.notmasked_contiguous` and `np.ma.flatnotmasked_contiguous` always return lists

This is the documented behavior, but previously the result could be any of slice, None, or list.

All downstream users seem to check for the `None` result from `flatnotmasked_contiguous` and replace it with `[]`. Those callers will continue to work as before.

`np.squeeze` restores old behavior of objects that cannot handle an `axis` argument

Prior to version `1.7.0`, `numpy.squeeze` did not have an `axis` argument and all empty axes were removed by default. The incorporation of an `axis` argument made it possible to selectively squeeze single or multiple empty axes, but the old API expectation was not respected because axes could still be selectively removed (silent success) from an object expecting all empty axes to be removed. That silent, selective removal of empty axes for objects expecting the old behavior has been fixed and the old behavior restored.

unstructured void array's `.item` method now returns a bytes object

`.item` now returns a `bytes` object instead of a buffer or byte array. This may affect code which assumed the return value was mutable, which is no longer the case.

`copy.copy` and `copy.deepcopy` no longer turn `masked` into an array

Since `np.ma.masked` is a readonly scalar, copying should be a no-op. These functions now behave consistently with `np.copy()`.

Multifield Indexing of Structured Arrays will still return a copy

The change that multi-field indexing of structured arrays returns a view instead of a copy is pushed back to 1.16. A new method `numpy.lib.recfunctions.repack_fields` has been introduced to help mitigate the effects of this change, which can be used to write code compatible with both numpy 1.15 and 1.16. For more information on how to update code to account for this future change see the "accessing multiple fields" section of the user guide.

15.47.6 C API changes

New functions `npy_get_floatstatus_barrier` and `npy_clear_floatstatus_barrier`

Functions `npy_get_floatstatus_barrier` and `npy_clear_floatstatus_barrier` have been added and should be used in place of the `npy_get_floatstatus``and ``npy_clear_status` functions. Optimizing compilers like GCC 8.1 and Clang were rearranging the order of operations when the previous functions were used in the ufunc SIMD functions, resulting in the floatstatus flags being checked before the operation whose status we wanted to check was run. See #10339.

Changes to `PyArray_GetDTypeTransferFunction`

`PyArray_GetDTypeTransferFunction` now defaults to using user-defined `copyswapn` / `copyswap` for user-defined dtypes. If this causes a significant performance hit, consider implementing `copyswapn` to reflect the implementation of `PyArray_GetStridedCopyFn`. See #10898.

15.47.7 New Features

`np.gcd` and `np.lcm` ufuncs added for integer and objects types

These compute the greatest common divisor, and lowest common multiple, respectively. These work on all the numpy integer types, as well as the builtin arbitrary-precision `Decimal` and `long` types.

Support for cross-platform builds for iOS

The build system has been modified to add support for the `_PYTHON_HOST_PLATFORM` environment variable, used by `distutils` when compiling on one platform for another platform. This makes it possible to compile NumPy for iOS targets.

This only enables you to compile NumPy for one specific platform at a time. Creating a full iOS-compatible NumPy package requires building for the 5 architectures supported by iOS (i386, x86_64, armv7, armv7s and arm64), and combining these 5 compiled builds products into a single "fat" binary.

`return_indices` keyword added for `np.intersect1d`

New keyword `return_indices` returns the indices of the two input arrays that correspond to the common elements.

`np.quantile` and `np.nanquantile`

Like `np.percentile` and `np.nanpercentile`, but takes quantiles in [0, 1] rather than percentiles in [0, 100]. `np.percentile` is now a thin wrapper around `np.quantile` with the extra step of dividing by 100.

Build system

Added experimental support for the 64-bit RISC-V architecture.

15.47.8 Improvements

`np.einsum` updates

Syncs einsum path optimization tech between `numpy` and *opt_einsum*. In particular, the *greedy* path has received many enhancements by @jcmgray. A full list of issues fixed are:

- Arbitrary memory can be passed into the *greedy* path. Fixes gh-11210.
- The greedy path has been updated to contain more dynamic programming ideas preventing a large number of duplicate (and expensive) calls that figure out the actual pair contraction that takes place. Now takes a few seconds on several hundred input tensors. Useful for matrix product state theories.
- Reworks the broadcasting dot error catching found in gh-11218 gh-10352 to be a bit earlier in the process.
- Enhances the *can_dot* functionality that previous missed an edge case (part of gh-11308).

`np.ufunc.reduce` and related functions now accept an initial value

`np.ufunc.reduce`, `np.sum`, `np.prod`, `np.min` and `np.max` all now accept an `initial` keyword argument that specifies the value to start the reduction with.

`np.flip` can operate over multiple axes

`np.flip` now accepts None, or tuples of int, in its `axis` argument. If axis is None, it will flip over all the axes.

`histogram` and `histogramdd` functions have moved to `np.lib.histograms`

These were originally found in `np.lib.function_base`. They are still available under their unscoped `np.histogram(dd)` names, and to maintain compatibility, aliased at `np.lib.function_base.histogram(dd)`.

Code that does `from np.lib.function_base import *` will need to be updated with the new location, and should consider not using `import *` in future.

`histogram` will accept NaN values when explicit bins are given

Previously it would fail when trying to compute a finite range for the data. Since the range is ignored anyway when the bins are given explicitly, this error was needless.

Note that calling `histogram` on NaN values continues to raise the `RuntimeWarning` s typical of working with nan values, which can be silenced as usual with `errstate`.

`histogram` works on datetime types, when explicit bin edges are given

Dates, times, and timedeltas can now be histogrammed. The bin edges must be passed explicitly, and are not yet computed automatically.

`histogram` "auto" estimator handles limited variance better

No longer does an IQR of 0 result in `n_bins=1`, rather the number of bins chosen is related to the data size in this situation.

The edges returned by *histogram*' and `histogramdd` now match the data float type

When passed `np.float16`, `np.float32`, or `np.longdouble` data, the returned edges are now of the same dtype. Previously, `histogram` would only return the same type if explicit bins were given, and `histogram` would produce `float64` bins no matter what the inputs.

`histogramdd` allows explicit ranges to be given in a subset of axes

The `range` argument of `numpy.histogramdd` can now contain `None` values to indicate that the range for the corresponding axis should be computed from the data. Previously, this could not be specified on a per-axis basis.

The normed arguments of `histogramdd` and `histogram2d` have been renamed

These arguments are now called `density`, which is consistent with `histogram`. The old argument continues to work, but the new name should be preferred.

`np.r_` works with 0d arrays, and `np.ma.mr_` works with `np.ma.masked`

0d arrays passed to the *r_* and *mr_* concatenation helpers are now treated as though they are arrays of length 1. Previously, passing these was an error. As a result, `numpy.ma.mr_` now works correctly on the `masked` constant.

`np.ptp` accepts a `keepdims` argument, and extended axis tuples

`np.ptp` (peak-to-peak) can now work over multiple axes, just like `np.max` and `np.min`.

`MaskedArray.astype` now is identical to `ndarray.astype`

This means it takes all the same arguments, making more code written for ndarray work for masked array too.

Enable AVX2/AVX512 at compile time

Change to simd.inc.src to allow use of AVX2 or AVX512 at compile time. Previously compilation for avx2 (or 512) with -march=native would still use the SSE code for the simd functions even when the rest of the code got AVX2.

`nan_to_num` always returns scalars when receiving scalar or 0d inputs

Previously an array was returned for integer scalar inputs, which is inconsistent with the behavior for float inputs, and that of ufuncs in general. For all types of scalar or 0d input, the result is now a scalar.

`np.flatnonzero` works on numpy-convertible types

`np.flatnonzero` now uses `np.ravel(a)` instead of `a.ravel()`, so it works for lists, tuples, etc.

`np.interp` returns numpy scalars rather than builtin scalars

Previously `np.interp(0.5, [0, 1], [10, 20])` would return a `float`, but now it returns a `np.float64` object, which more closely matches the behavior of other functions.

Additionally, the special case of `np.interp(object_array_0d, ...)` is no longer supported, as `np.interp(object_array_nd)` was never supported anyway.

As a result of this change, the `period` argument can now be used on 0d arrays.

Allow dtype field names to be unicode in Python 2

Previously `np.dtype([(u'name', float)])` would raise a `TypeError` in Python 2, as only bytestrings were allowed in field names. Now any unicode string field names will be encoded with the `ascii` codec, raising a `UnicodeEncodeError` upon failure.

This change makes it easier to write Python 2/3 compatible code using `from __future__ import unicode_literals`, which previously would cause string literal field names to raise a TypeError in Python 2.

Comparison ufuncs accept `dtype=object`, overriding the default `bool`

This allows object arrays of symbolic types, which override == and other operators to return expressions, to be compared elementwise with `np.equal(a, b, dtype=object)`.

`sort` functions accept `kind='stable'`

Up until now, to perform a stable sort on the data, the user must do:

```
>>> np.sort([5, 2, 6, 2, 1], kind='mergesort')
[1, 2, 2, 5, 6]
```

because merge sort is the only stable sorting algorithm available in NumPy. However, having kind='mergesort' does not make it explicit that the user wants to perform a stable sort thus harming the readability.

This change allows the user to specify kind='stable' thus clarifying the intent.

Do not make temporary copies for in-place accumulation

When ufuncs perform accumulation they no longer make temporary copies because of the overlap between input an output, that is, the next element accumulated is added before the accumulated result is stored in its place, hence the overlap is safe. Avoiding the copy results in faster execution.

`linalg.matrix_power` can now handle stacks of matrices

Like other functions in `linalg`, `matrix_power` can now deal with arrays of dimension larger than 2, which are treated as stacks of matrices. As part of the change, to further improve consistency, the name of the first argument has been changed to `a` (from `M`), and the exceptions for non-square matrices have been changed to `LinAlgError` (from `ValueError`).

Increased performance in `random.permutation` for multidimensional arrays

`permutation` uses the fast path in `random.shuffle` for all input array dimensions. Previously the fast path was only used for 1-d arrays.

Generalized ufuncs now accept `axes`, `axis` and `keepdims` arguments

One can control over which axes a generalized ufunc operates by passing in an `axes` argument, a list of tuples with indices of particular axes. For instance, for a signature of `(i,j),(j,k)->(i,k)` appropriate for matrix multiplication, the base elements are two-dimensional matrices and these are taken to be stored in the two last axes of each argument. The corresponding axes keyword would be `[(-2, -1), (-2, -1), (-2, -1)]`. If one wanted to use leading dimensions instead, one would pass in `[(0, 1), (0, 1), (0, 1)]`.

For simplicity, for generalized ufuncs that operate on 1-dimensional arrays (vectors), a single integer is accepted instead of a single-element tuple, and for generalized ufuncs for which all outputs are scalars, the (empty) output tuples can be omitted. Hence, for a signature of `(i),(i)->()` appropriate for an inner product, one could pass in `axes=[0, 0]` to indicate that the vectors are stored in the first dimensions of the two inputs arguments.

As a short-cut for generalized ufuncs that are similar to reductions, i.e., that act on a single, shared core dimension such as the inner product example above, one can pass an `axis` argument. This is equivalent to passing in `axes` with identical entries for all arguments with that core dimension (e.g., for the example above, `axes=[(axis,), (axis,)]`).

Furthermore, like for reductions, for generalized ufuncs that have inputs that all have the same number of core dimensions and outputs with no core dimension, one can pass in `keepdims` to leave a dimension with size 1 in the outputs, thus allowing proper broadcasting against the original inputs. The location of the extra dimension can be controlled with `axes`. For instance, for the inner-product example, `keepdims=True, axes=[-2, -2, -2]` would act on the inner-product example, `keepdims=True, axis=-2` would act on the one-but-last dimension of the input arguments, and leave a size 1 dimension in that place in the output.

float128 values now print correctly on ppc systems

Previously printing float128 values was buggy on ppc, since the special double-double floating-point-format on these systems was not accounted for. float128s now print with correct rounding and uniqueness.

Warning to ppc users: You should upgrade glibc if it is version <=2.23, especially if using float128. On ppc, glibc's malloc in these version often misaligns allocated memory which can crash numpy when using float128 values.

New `np.take_along_axis` and `np.put_along_axis` functions

When used on multidimensional arrays, `argsort`, `argmin`, `argmax`, and `argpartition` return arrays that are difficult to use as indices. `take_along_axis` provides an easy way to use these indices to lookup values within an array, so that:

```
np.take_along_axis(a, np.argsort(a, axis=axis), axis=axis)
```

is the same as:

```
np.sort(a, axis=axis)
```

`np.put_along_axis` acts as the dual operation for writing to these indices within an array.

15.48 NumPy 1.14.6 Release Notes

This is a bugfix release for bugs reported following the 1.14.5 release. The most significant fixes are:

- Fix for behavior change in `ma.masked_values(shrink=True)`
- Fix the new cached allocations machinery to be thread safe.

The Python versions supported in this release are 2.7 and 3.4 - 3.7. The Python 3.6 wheels on PyPI should be compatible with all Python 3.6 versions.

15.48.1 Contributors

A total of 4 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris
- Eric Wieser
- Julian Taylor
- Matti Picus

15.48.2 Pull requests merged

A total of 4 pull requests were merged for this release.

- #11985: BUG: fix cached allocations without the GIL
- #11986: BUG: Undo behavior change in ma.masked_values(shrink=True)
- #11987: BUG: fix refcount leak in PyArray_AdaptFlexibleDType
- #11995: TST: Add Python 3.7 testing to NumPy 1.14.

15.49 NumPy 1.14.5 Release Notes

This is a bugfix release for bugs reported following the 1.14.4 release. The most significant fixes are:

- fixes for compilation errors on alpine and NetBSD

The Python versions supported in this release are 2.7 and 3.4 - 3.6. The Python 3.6 wheels available from PIP are built with Python 3.6.2 and should be compatible with all previous versions of Python 3.6. The source releases were cythonized with Cython 0.28.2 and should work for the upcoming Python 3.7.

15.49.1 Contributors

A total of 1 person contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris

15.49.2 Pull requests merged

A total of 2 pull requests were merged for this release.

- #11274: BUG: Correct use of NPY_UNUSED.
- #11294: BUG: Remove extra trailing parentheses.

15.50 NumPy 1.14.4 Release Notes

This is a bugfix release for bugs reported following the 1.14.3 release. The most significant fixes are:

- fixes for compiler instruction reordering that resulted in NaN's not being properly propagated in *np.max* and *np.min*,
- fixes for bus faults on SPARC and older ARM due to incorrect alignment checks.

There are also improvements to printing of long doubles on PPC platforms. All is not yet perfect on that platform, the whitespace padding is still incorrect and is to be fixed in numpy 1.15, consequently NumPy still fails some printing-related (and other) unit tests on ppc systems. However, the printed values are now correct.

Note that NumPy will error on import if it detects incorrect float32 *dot* results. This problem has been seen on the Mac when working in the Anaconda environment and is due to a subtle interaction between MKL and PyQt5. It is not strictly a NumPy problem, but it is best that users be aware of it. See the gh-8577 NumPy issue for more information.

The Python versions supported in this release are 2.7 and 3.4 - 3.6. The Python 3.6 wheels available from PIP are built with Python 3.6.2 and should be compatible with all previous versions of Python 3.6. The source releases were cythonized with Cython 0.28.2 and should work for the upcoming Python 3.7.

15.50.1 Contributors

A total of 7 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Allan Haldane
- Charles Harris
- Marten van Kerkwijk
- Matti Picus

- Pauli Virtanen
- Ryan Soklaski +
- Sebastian Berg

15.50.2 Pull requests merged

A total of 11 pull requests were merged for this release.

- #11104: BUG: str of DOUBLE_DOUBLE format wrong on ppc64
- #11170: TST: linalg: add regression test for gh-8577
- #11174: MAINT: add sanity-checks to be run at import time
- #11181: BUG: void dtype setup checked offset not actual pointer for alignment
- #11194: BUG: Python2 doubles don't print correctly in interactive shell.
- #11198: BUG: optimizing compilers can reorder call to npy_get_floatstatus
- #11199: BUG: reduce using SSE only warns if inside SSE loop
- #11203: BUG: Bytes delimiter/comments in genfromtxt should be decoded
- #11211: BUG: Fix reference count/memory leak exposed by better testing
- #11219: BUG: Fixes einsum broadcasting bug when optimize=True
- #11251: DOC: Document 1.14.4 release.

15.51 NumPy 1.14.3 Release Notes

This is a bugfix release for a few bugs reported following the 1.14.2 release:

- np.lib.recfunctions.fromrecords accepts a list-of-lists, until 1.15
- In python2, float types use the new print style when printing to a file
- style arg in "legacy" print mode now works for 0d arrays

The Python versions supported in this release are 2.7 and 3.4 - 3.6. The Python 3.6 wheels available from PIP are built with Python 3.6.2 and should be compatible with all previous versions of Python 3.6. The source releases were cythonized with Cython 0.28.2.

15.51.1 Contributors

A total of 6 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Allan Haldane
- Charles Harris
- Jonathan March +
- Malcolm Smith +
- Matti Picus
- Pauli Virtanen

15.51.2 Pull requests merged

A total of 8 pull requests were merged for this release.

- #10862: BUG: floating types should override tp_print (1.14 backport)
- #10905: BUG: for 1.14 back-compat, accept list-of-lists in fromrecords
- #10947: BUG: 'style' arg to array2string broken in legacy mode (1.14…
- #10959: BUG: test, fix for missing flags['WRITEBACKIFCOPY'] key
- #10960: BUG: Add missing underscore to prototype in check_embedded_lapack
- #10961: BUG: Fix encoding regression in ma/bench.py (Issue #10868)
- #10962: BUG: core: fix NPY_TITLE_KEY macro on pypy
- #10974: BUG: test, fix PyArray_DiscardWritebackIfCopy…

15.52 NumPy 1.14.2 Release Notes

This is a bugfix release for some bugs reported following the 1.14.1 release. The major problems dealt with are as follows.

- Residual bugs in the new array printing functionality.
- Regression resulting in a relocation problem with shared library.
- Improved PyPy compatibility.

The Python versions supported in this release are 2.7 and 3.4 - 3.6. The Python 3.6 wheels available from PIP are built with Python 3.6.2 and should be compatible with all previous versions of Python 3.6. The source releases were cythonized with Cython 0.26.1, which is known to **not** support the upcoming Python 3.7 release. People who wish to run Python 3.7 should check out the NumPy repo and try building with the, as yet, unreleased master branch of Cython.

15.52.1 Contributors

A total of 4 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Allan Haldane
- Charles Harris
- Eric Wieser
- Pauli Virtanen

15.52.2 Pull requests merged

A total of 5 pull requests were merged for this release.

- #10674: BUG: Further back-compat fix for subclassed array repr
- #10725: BUG: dragon4 fractional output mode adds too many trailing zeros
- #10726: BUG: Fix f2py generated code to work on PyPy
- #10727: BUG: Fix missing NPY_VISIBILITY_HIDDEN on npy_longdouble_to_PyLong
- #10729: DOC: Create 1.14.2 notes and changelog.

15.53 NumPy 1.14.1 Release Notes

This is a bugfix release for some problems reported following the 1.14.0 release. The major problems fixed are the following.

- Problems with the new array printing, particularly the printing of complex values, Please report any additional problems that may turn up.
- Problems with `np.einsum` due to the new `optimized=True` default. Some fixes for optimization have been applied and `optimize=False` is now the default.
- The sort order in `np.unique` when `axis=<some-number>` will now always be lexicographic in the subarray elements. In previous NumPy versions there was an optimization that could result in sorting the subarrays as unsigned byte strings.
- The change in 1.14.0 that multi-field indexing of structured arrays returns a view instead of a copy has been reverted but remains on track for NumPy 1.15. Affected users should read the 1.14.1 Numpy User Guide section "basics/structured arrays/accessing multiple fields" for advice on how to manage this transition.

The Python versions supported in this release are 2.7 and 3.4 - 3.6. The Python 3.6 wheels available from PIP are built with Python 3.6.2 and should be compatible with all previous versions of Python 3.6. The source releases were cythonized with Cython 0.26.1, which is known to **not** support the upcoming Python 3.7 release. People who wish to run Python 3.7 should check out the NumPy repo and try building with the, as yet, unreleased master branch of Cython.

15.53.1 Contributors

A total of 14 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Allan Haldane
- Charles Harris
- Daniel Smith
- Dennis Weyland +
- Eric Larson
- Eric Wieser
- Jarrod Millman
- Kenichi Maehashi +
- Marten van Kerkwijk
- Mathieu Lamarre
- Sebastian Berg
- Simon Conseil
- Simon Gibbons
- xoviat

15.53.2 Pull requests merged

A total of 36 pull requests were merged for this release.

- #10339: BUG: restrict the __config__ modifications to win32
- #10368: MAINT: Adjust type promotion in linalg.norm
- #10375: BUG: add missing paren and remove quotes from repr of fieldless…
- #10395: MAINT: Update download URL in setup.py.
- #10396: BUG: fix einsum issue with unicode input and py2
- #10397: BUG: fix error message not formatted in einsum
- #10398: DOC: add documentation about how to handle new array printing
- #10403: BUG: Set einsum optimize parameter default to *False*.
- #10424: ENH: Fix repr of np.record objects to match np.void types #10412
- #10425: MAINT: Update zesty to artful for i386 testing
- #10431: REL: Add 1.14.1 release notes template
- #10435: MAINT: Use ValueError for duplicate field names in lookup (backport)
- #10534: BUG: Provide a better error message for out-of-order fields
- #10536: BUG: Resize bytes columns in genfromtxt (backport of #10401)
- #10537: BUG: multifield-indexing adds padding bytes: revert for 1.14.1
- #10539: BUG: fix np.save issue with python 2.7.5
- #10540: BUG: Add missing DECREF in Py2 int() cast
- #10541: TST: Add circleci document testing to maintenance/1.14.x
- #10542: BUG: complex repr has extra spaces, missing + (1.14 backport)
- #10550: BUG: Set missing exception after malloc
- #10557: BUG: In numpy.i, clear CARRAY flag if wrapped buffer is not C_CONTIGUOUS.
- #10558: DEP: Issue FutureWarning when malformed records detected.
- #10559: BUG: Fix einsum optimize logic for singleton dimensions
- #10560: BUG: Fix calling ufuncs with a positional output argument.
- #10561: BUG: Fix various Big-Endian test failures (ppc64)
- #10562: BUG: Make dtype.descr error for out-of-order fields.
- #10563: BUG: arrays not being flattened in *union1d*
- #10607: MAINT: Update sphinxext submodule hash.
- #10608: BUG: Revert sort optimization in np.unique.
- #10609: BUG: infinite recursion in str of 0d subclasses
- #10610: BUG: Align type definition with generated lapack
- #10612: BUG/ENH: Improve output for structured non-void types
- #10622: BUG: deallocate recursive closure in arrayprint.py (1.14 backport)
- #10624: BUG: Correctly identify comma separated dtype strings

- #10629: BUG: deallocate recursive closure in arrayprint.py (backport…
- #10630: REL: Prepare for 1.14.1 release.

15.54 NumPy 1.14.0 Release Notes

Numpy 1.14.0 is the result of seven months of work and contains a large number of bug fixes and new features, along with several changes with potential compatibility issues. The major change that users will notice are the stylistic changes in the way numpy arrays and scalars are printed, a change that will affect doctests. See below for details on how to preserve the old style printing when needed.

A major decision affecting future development concerns the schedule for dropping Python 2.7 support in the runup to 2020. The decision has been made to support 2.7 for all releases made in 2018, with the last release being designated a long term release with support for bug fixes extending through 2019. In 2019 support for 2.7 will be dropped in all new releases. More details can be found in NEP 12.

This release supports Python 2.7 and 3.4 - 3.6.

15.54.1 Highlights

- The *np.einsum* function uses BLAS when possible
- `genfromtxt`, `loadtxt`, `fromregex` and `savetxt` can now handle files with arbitrary Python supported encoding.
- Major improvements to printing of NumPy arrays and scalars.

15.54.2 New functions

- `parametrize`: decorator added to numpy.testing
- `chebinterpolate`: Interpolate function at Chebyshev points.
- `format_float_positional` and `format_float_scientific` : format floating-point scalars unambiguously with control of rounding and padding.
- `PyArray_ResolveWritebackIfCopy` and `PyArray_SetWritebackIfCopyBase`, new C-API functions useful in achieving PyPy compatibility.

15.54.3 Deprecations

- Using `np.bool_` objects in place of integers is deprecated. Previously `operator.index(np.bool_)` was legal and allowed constructs such as `[1, 2, 3][np.True_]`. That was misleading, as it behaved differently from `np.array([1, 2, 3])[np.True_]`.
- Truth testing of an empty array is deprecated. To check if an array is not empty, use `array.size > 0`.
- Calling `np.bincount` with `minlength=None` is deprecated. `minlength=0` should be used instead.
- Calling `np.fromstring` with the default value of the `sep` argument is deprecated. When that argument is not provided, a broken version of `np.frombuffer` is used that silently accepts unicode strings and – after encoding them as either utf-8 (python 3) or the default encoding (python 2) – treats them as binary data. If reading binary data is desired, `np.frombuffer` should be used directly.
- The `style` option of array2string is deprecated in non-legacy printing mode.

- `PyArray_SetUpdateIfCopyBase` has been deprecated. For NumPy versions >= 1.14 use `PyArray_SetWritebackIfCopyBase` instead, see *C API changes* below for more details.
- The use of `UPDATEIFCOPY` arrays is deprecated, see *C API changes* below for details. We will not be dropping support for those arrays, but they are not compatible with PyPy.

15.54.4 Future Changes

- `np.issubdtype` will stop downcasting dtype-like arguments. It might be expected that `issubdtype(np.float32, 'float64')` and `issubdtype(np.float32, np.float64)` mean the same thing - however, there was an undocumented special case that translated the former into `issubdtype(np.float32, np.floating)`, giving the surprising result of True.

  This translation now gives a warning that explains what translation is occurring. In the future, the translation will be disabled, and the first example will be made equivalent to the second.
- `np.linalg.lstsq` default for `rcond` will be changed. The `rcond` parameter to `np.linalg.lstsq` will change its default to machine precision times the largest of the input array dimensions. A FutureWarning is issued when `rcond` is not passed explicitly.
- `a.flat.__array__()` will return a writeable copy of `a` when `a` is non-contiguous. Previously it returned an UPDATEIFCOPY array when `a` was writeable. Currently it returns a non-writeable copy. See gh-7054 for a discussion of the issue.
- Unstructured void array's `.item` method will return a bytes object. In the future, calling `.item()` on arrays or scalars of `np.void` datatype will return a `bytes` object instead of a buffer or int array, the same as returned by `bytes(void_scalar)`. This may affect code which assumed the return value was mutable, which will no longer be the case. A `FutureWarning` is now issued when this would occur.

15.54.5 Compatibility notes

The mask of a masked array view is also a view rather than a copy

There was a FutureWarning about this change in NumPy 1.11.x. In short, it is now the case that, when changing a view of a masked array, changes to the mask are propagated to the original. That was not previously the case. This change affects slices in particular. Note that this does not yet work properly if the mask of the original array is `nomask` and the mask of the view is changed. See gh-5580 for an extended discussion. The original behavior of having a copy of the mask can be obtained by calling the `unshare_mask` method of the view.

`np.ma.masked` is no longer writeable

Attempts to mutate the `masked` constant now error, as the underlying arrays are marked readonly. In the past, it was possible to get away with:

```
# emulating a function that sometimes returns np.ma.masked
val = random.choice([np.ma.masked, 10])
var_arr = np.asarray(val)
val_arr += 1  # now errors, previously changed np.ma.masked.data
```

`np.ma` functions producing `fill_value` s have changed

Previously, `np.ma.default_fill_value` would return a 0d array, but `np.ma.minimum_fill_value` and `np.ma.maximum_fill_value` would return a tuple of the fields. Instead, all three methods return a structured `np.void` object, which is what you would already find in the `.fill_value` attribute.

Additionally, the dtype guessing now matches that of `np.array` - so when passing a python scalar `x`, `maximum_fill_value(x)` is always the same as `maximum_fill_value(np.array(x))`. Previously `x = long(1)` on Python 2 violated this assumption.

`a.flat.__array__()` returns non-writeable arrays when `a` is non-contiguous

The intent is that the UPDATEIFCOPY array previously returned when `a` was non-contiguous will be replaced by a writeable copy in the future. This temporary measure is aimed to notify folks who expect the underlying array be modified in this situation that that will no longer be the case. The most likely places for this to be noticed is when expressions of the form `np.asarray(a.flat)` are used, or when `a.flat` is passed as the out parameter to a ufunc.

`np.tensordot` now returns zero array when contracting over 0-length dimension

Previously `np.tensordot` raised a ValueError when contracting over 0-length dimension. Now it returns a zero array, which is consistent with the behaviour of `np.dot` and `np.einsum`.

`numpy.testing` reorganized

This is not expected to cause problems, but possibly something has been left out. If you experience an unexpected import problem using `numpy.testing` let us know.

`np.asfarray` no longer accepts non-dtypes through the `dtype` argument

This previously would accept `dtype=some_array`, with the implied semantics of `dtype=some_array.dtype`. This was undocumented, unique across the numpy functions, and if used would likely correspond to a typo.

1D `np.linalg.norm` preserves float input types, even for arbitrary orders

Previously, this would promote to `float64` when arbitrary orders were passed, despite not doing so under the simple cases:

```
>>> f32 = np.float32([[1, 2]])
>>> np.linalg.norm(f32, 2.0, axis=-1).dtype
dtype('float32')
>>> np.linalg.norm(f32, 2.0001, axis=-1).dtype
dtype('float64')  # numpy 1.13
dtype('float32')  # numpy 1.14
```

This change affects only `float32` and `float16` arrays.

`count_nonzero(arr, axis=())` now counts over no axes, not all axes

Elsewhere, `axis==()` is always understood as "no axes", but *count_nonzero* had a special case to treat this as "all axes". This was inconsistent and surprising. The correct way to count over all axes has always been to pass `axis == None`.

`__init__.py` files added to test directories

This is for pytest compatibility in the case of duplicate test file names in the different directories. As a result, `run_module_suite` no longer works, i.e., `python <path-to-test-file>` results in an error.

`.astype(bool)` on unstructured void arrays now calls `bool` on each element

On Python 2, `void_array.astype(bool)` would always return an array of `True`, unless the dtype is `V0`. On Python 3, this operation would usually crash. Going forwards, *astype* matches the behavior of `bool(np.void)`, considering a buffer of all zeros as false, and anything else as true. Checks for `V0` can still be done with `arr.dtype.itemsize == 0`.

`MaskedArray.squeeze` never returns `np.ma.masked`

`np.squeeze` is documented as returning a view, but the masked variant would sometimes return `masked`, which is not a view. This has been fixed, so that the result is always a view on the original masked array. This breaks any code that used `masked_arr.squeeze() is np.ma.masked`, but fixes code that writes to the result of *squeeze()*.

Renamed first parameter of `can_cast` from `from` to `from_`

The previous parameter name `from` is a reserved keyword in Python, which made it difficult to pass the argument by name. This has been fixed by renaming the parameter to `from_`.

`isnat` raises `TypeError` when passed wrong type

The ufunc `isnat` used to raise a `ValueError` when it was not passed variables of type `datetime` or `timedelta`. This has been changed to raising a `TypeError`.

`dtype.__getitem__` raises `TypeError` when passed wrong type

When indexed with a float, the dtype object used to raise `ValueError`.

User-defined types now need to implement `__str__` and `__repr__`

Previously, user-defined types could fall back to a default implementation of `__str__` and `__repr__` implemented in numpy, but this has now been removed. Now user-defined types will fall back to the python default `object.__str__` and `object.__repr__`.

Many changes to array printing, disableable with the new "legacy" printing mode

The `str` and `repr` of ndarrays and numpy scalars have been changed in a variety of ways. These changes are likely to break downstream user's doctests.

These new behaviors can be disabled to mostly reproduce numpy 1.13 behavior by enabling the new 1.13 "legacy" printing mode. This is enabled by calling `np.set_printoptions(legacy="1.13")`, or using the new `legacy` argument to `np.array2string`, as `np.array2string(arr, legacy='1.13')`.

In summary, the major changes are:

- For floating-point types:
  - The `repr` of float arrays often omits a space previously printed in the sign position. See the new `sign` option to `np.set_printoptions`.
  - Floating-point arrays and scalars use a new algorithm for decimal representations, giving the shortest unique representation. This will usually shorten `float16` fractional output, and sometimes `float32` and `float128` output. `float64` should be unaffected. See the new `floatmode` option to `np.set_printoptions`.
  - Float arrays printed in scientific notation no longer use fixed-precision, and now instead show the shortest unique representation.
  - The `str` of floating-point scalars is no longer truncated in python2.
- For other data types:
  - Non-finite complex scalars print like `nanj` instead of `nan*j`.
  - `NaT` values in datetime arrays are now properly aligned.
  - Arrays and scalars of `np.void` datatype are now printed using hex notation.
- For line-wrapping:
  - The "dtype" part of ndarray reprs will now be printed on the next line if there isn't space on the last line of array output.
  - The `linewidth` format option is now always respected. The *repr* or *str* of an array will never exceed this, unless a single element is too wide.
  - The last line of an array string will never have more elements than earlier lines.
  - An extra space is no longer inserted on the first line if the elements are too wide.
- For summarization (the use of `...` to shorten long arrays):
  - A trailing comma is no longer inserted for `str`. Previously, `str(np.arange(1001))` gave `'[ 0 1 2 ..., 998 999 1000]'`, which has an extra comma.
  - For arrays of 2-D and beyond, when `...` is printed on its own line in order to summarize any but the last axis, newlines are now appended to that line to match its leading newlines and a trailing space character is removed.
- `MaskedArray` arrays now separate printed elements with commas, always print the dtype, and correctly wrap the elements of long arrays to multiple lines. If there is more than 1 dimension, the array attributes are now printed in a new "left-justified" printing style.
- `recarray` arrays no longer print a trailing space before their dtype, and wrap to the right number of columns.
- 0d arrays no longer have their own idiosyncratic implementations of `str` and `repr`. The `style` argument to `np.array2string` is deprecated.
- Arrays of `bool` datatype will omit the datatype in the `repr`.

- User-defined `dtypes` (subclasses of `np.generic`) now need to implement `__str__` and `__repr__`.

Some of these changes are described in more detail below. If you need to retain the previous behavior for doctests or other reasons, you may want to do something like:

```
# FIXME: We need the str/repr formatting used in Numpy < 1.14.
try:
    np.set_printoptions(legacy='1.13')
except TypeError:
    pass
```

15.54.6 C API changes

PyPy compatible alternative to `UPDATEIFCOPY` arrays

`UPDATEIFCOPY` arrays are contiguous copies of existing arrays, possibly with different dimensions, whose contents are copied back to the original array when their refcount goes to zero and they are deallocated. Because PyPy does not use refcounts, they do not function correctly with PyPy. NumPy is in the process of eliminating their use internally and two new C-API functions,

- `PyArray_SetWritebackIfCopyBase`
- `PyArray_ResolveWritebackIfCopy`,

have been added together with a complementary flag, `NPY_ARRAY_WRITEBACKIFCOPY`. Using the new functionality also requires that some flags be changed when new arrays are created, to wit: `NPY_ARRAY_INOUT_ARRAY` should be replaced by `NPY_ARRAY_INOUT_ARRAY2` and `NPY_ARRAY_INOUT_FARRAY` should be replaced by `NPY_ARRAY_INOUT_FARRAY2`. Arrays created with these new flags will then have the `WRITEBACKIFCOPY` semantics.

If PyPy compatibility is not a concern, these new functions can be ignored, although there will be a `DeprecationWarning`. If you do wish to pursue PyPy compatibility, more information on these functions and their use may be found in the c-api documentation and the example in how-to-extend.

15.54.7 New Features

Encoding argument for text IO functions

`genfromtxt`, `loadtxt`, `fromregex` and `savetxt` can now handle files with arbitrary encoding supported by Python via the encoding argument. For backward compatibility the argument defaults to the special `bytes` value which continues to treat text as raw byte values and continues to pass latin1 encoded bytes to custom converters. Using any other value (including `None` for system default) will switch the functions to real text IO so one receives unicode strings instead of bytes in the resulting arrays.

External `nose` plugins are usable by `numpy.testing.Tester`

`numpy.testing.Tester` is now aware of `nose` plugins that are outside the `nose` built-in ones. This allows using, for example, `nose-timer` like so: `np.test(extra_argv=['--with-timer', '--timer-top-n', '20'])` to obtain the runtime of the 20 slowest tests. An extra keyword `timer` was also added to `Tester.test`, so `np.test(timer=20)` will also report the 20 slowest tests.

`parametrize` decorator added to `numpy.testing`

A basic `parametrize` decorator is now available in `numpy.testing`. It is intended to allow rewriting yield based tests that have been deprecated in pytest so as to facilitate the transition to pytest in the future. The nose testing framework has not been supported for several years and looks like abandonware.

The new `parametrize` decorator does not have the full functionality of the one in pytest. It doesn't work for classes, doesn't support nesting, and does not substitute variable names. Even so, it should be adequate to rewrite the NumPy tests.

`chebinterpolate` function added to `numpy.polynomial.chebyshev`

The new `chebinterpolate` function interpolates a given function at the Chebyshev points of the first kind. A new `Chebyshev.interpolate` class method adds support for interpolation over arbitrary intervals using the scaled and shifted Chebyshev points of the first kind.

Support for reading lzma compressed text files in Python 3

With Python versions containing the `lzma` module the text IO functions can now transparently read from files with `xz` or `lzma` extension.

`sign` option added to `np.setprintoptions` and `np.array2string`

This option controls printing of the sign of floating-point types, and may be one of the characters '-', '+' or ' '. With '+' numpy always prints the sign of positive values, with ' ' it always prints a space (whitespace character) in the sign position of positive values, and with '-' it will omit the sign character for positive values. The new default is '-'.

This new default changes the float output relative to numpy 1.13. The old behavior can be obtained in 1.13 "legacy" printing mode, see compatibility notes above.

`hermitian` option added to"np.linalg.matrix_rank"

The new `hermitian` option allows choosing between standard SVD based matrix rank calculation and the more efficient eigenvalue based method for symmetric/hermitian matrices.

`threshold` and `edgeitems` options added to `np.array2string`

These options could previously be controlled using `np.set_printoptions`, but now can be changed on a per-call basis as arguments to `np.array2string`.

`concatenate` and `stack` gained an `out` argument

A preallocated buffer of the desired dtype can now be used for the output of these functions.

Support for PGI flang compiler on Windows

The PGI flang compiler is a Fortran front end for LLVM released by NVIDIA under the Apache 2 license. It can be invoked by

```
python setup.py config --compiler=clang --fcompiler=flang install
```

There is little experience with this new compiler, so any feedback from people using it will be appreciated.

15.54.8 Improvements

Numerator degrees of freedom in `random.noncentral_f` need only be positive.

Prior to NumPy 1.14.0, the numerator degrees of freedom needed to be > 1, but the distribution is valid for values > 0, which is the new requirement.

The GIL is released for all `np.einsum` variations

Some specific loop structures which have an accelerated loop version did not release the GIL prior to NumPy 1.14.0. This oversight has been fixed.

The *np.einsum* function will use BLAS when possible and optimize by default

The `np.einsum` function will now call `np.tensordot` when appropriate. Because `np.tensordot` uses BLAS when possible, that will speed up execution. By default, `np.einsum` will also attempt optimization as the overhead is small relative to the potential improvement in speed.

`f2py` now handles arrays of dimension 0

`f2py` now allows for the allocation of arrays of dimension 0. This allows for more consistent handling of corner cases downstream.

`numpy.distutils` supports using MSVC and mingw64-gfortran together

Numpy distutils now supports using Mingw64 gfortran and MSVC compilers together. This enables the production of Python extension modules on Windows containing Fortran code while retaining compatibility with the binaries distributed by Python.org. Not all use cases are supported, but most common ways to wrap Fortran for Python are functional.

Compilation in this mode is usually enabled automatically, and can be selected via the `--fcompiler` and `--compiler` options to `setup.py`. Moreover, linking Fortran codes to static OpenBLAS is supported; by default a gfortran compatible static archive `openblas.a` is looked for.

`np.linalg.pinv` now works on stacked matrices

Previously it was limited to a single 2d array.

`numpy.save` aligns data to 64 bytes instead of 16

Saving NumPy arrays in the `npy` format with `numpy.save` inserts padding before the array data to align it at 64 bytes. Previously this was only 16 bytes (and sometimes less due to a bug in the code for version 2). Now the alignment is 64 bytes, which matches the widest SIMD instruction set commonly available, and is also the most common cache line size. This makes `npy` files easier to use in programs which open them with `mmap`, especially on Linux where an `mmap` offset must be a multiple of the page size.

NPZ files now can be written without using temporary files

In Python 3.6+ `numpy.savez` and `numpy.savez_compressed` now write directly to a ZIP file, without creating intermediate temporary files.

Better support for empty structured and string types

Structured types can contain zero fields, and string dtypes can contain zero characters. Zero-length strings still cannot be created directly, and must be constructed through structured dtypes:

```
str0 = np.empty(10, np.dtype([('v', str, N)]))['v']
void0 = np.empty(10, np.void)
```

It was always possible to work with these, but the following operations are now supported for these arrays:

- *arr.sort()*
- *arr.view(bytes)*
- *arr.resize(…)*
- *pickle.dumps(arr)*

Support for `decimal.Decimal` in `np.lib.financial`

Unless otherwise stated all functions within the `financial` package now support using the `decimal.Decimal` built-in type.

Float printing now uses "dragon4" algorithm for shortest decimal representation

The `str` and `repr` of floating-point values (16, 32, 64 and 128 bit) are now printed to give the shortest decimal representation which uniquely identifies the value from others of the same type. Previously this was only true for `float64` values. The remaining float types will now often be shorter than in numpy 1.13. Arrays printed in scientific notation now also use the shortest scientific representation, instead of fixed precision as before.

> Additionally, the *str* of float scalars scalars will no longer be truncated in python2, unlike python2 *float's*. *'np.double* scalars now have a `str` and `repr` identical to that of a python3 float.

New functions `np.format_float_scientific` and `np.format_float_positional` are provided to generate these decimal representations.

A new option `floatmode` has been added to `np.set_printoptions` and `np.array2string`, which gives control over uniqueness and rounding of printed elements in an array. The new default is `floatmode='maxprec'` with `precision=8`, which will print at most 8 fractional digits, or fewer if an element can be uniquely represented with fewer. A useful new mode is `floatmode="unique"`, which will output enough digits to specify the array elements uniquely.

Numpy complex-floating-scalars with values like `inf*j` or `nan*j` now print as `infj` and `nanj`, like the pure-python `complex` type.

The `FloatFormat` and `LongFloatFormat` classes are deprecated and should both be replaced by `FloatingFormat`. Similarly `ComplexFormat` and `LongComplexFormat` should be replaced by `ComplexFloatingFormat`.

`void` datatype elements are now printed in hex notation

A hex representation compatible with the python `bytes` type is now printed for unstructured `np.void` elements, e.g., `V4` datatype. Previously, in python2 the raw void data of the element was printed to stdout, or in python3 the integer byte values were shown.

printing style for `void` datatypes is now independently customizable

The printing style of `np.void` arrays is now independently customizable using the `formatter` argument to `np.set_printoptions`, using the `'void'` key, instead of the catch-all `numpystr` key as before.

Reduced memory usage of `np.loadtxt`

`np.loadtxt` now reads files in chunks instead of all at once which decreases its memory usage significantly for large files.

15.54.9 Changes

Multiple-field indexing/assignment of structured arrays

The indexing and assignment of structured arrays with multiple fields has changed in a number of ways, as warned about in previous releases.

First, indexing a structured array with multiple fields, e.g., `arr[['f1', 'f3']]`, returns a view into the original array instead of a copy. The returned view will have extra padding bytes corresponding to intervening fields in the original array, unlike the copy in 1.13, which will affect code such as `arr[['f1', 'f3']].view(newdtype)`.

Second, assignment between structured arrays will now occur "by position" instead of "by field name". The Nth field of the destination will be set to the Nth field of the source regardless of field name, unlike in numpy versions 1.6 to 1.13 in which fields in the destination array were set to the identically-named field in the source array or to 0 if the source did not have a field.

Correspondingly, the order of fields in a structured dtypes now matters when computing dtype equality. For example, with the dtypes

```
x = dtype({'names': ['A', 'B'], 'formats': ['i4', 'f4'], 'offsets': [0, 4]})
y = dtype({'names': ['B', 'A'], 'formats': ['f4', 'i4'], 'offsets': [4, 0]})
```

the expression `x == y` will now return `False`, unlike before. This makes dictionary based dtype specifications like `dtype({'a': ('i4', 0), 'b': ('f4', 4)})` dangerous in python < 3.6 since dict key order is not preserved in those versions.

Assignment from a structured array to a boolean array now raises a ValueError, unlike in 1.13, where it always set the destination elements to `True`.

Assignment from structured array with more than one field to a non-structured array now raises a ValueError. In 1.13 this copied just the first field of the source to the destination.

Using field "titles" in multiple-field indexing is now disallowed, as is repeating a field name in a multiple-field index.

The documentation for structured arrays in the user guide has been significantly updated to reflect these changes.

Integer and Void scalars are now unaffected by `np.set_string_function`

Previously, unlike most other numpy scalars, the `str` and `repr` of integer and void scalars could be controlled by `np.set_string_function`. This is no longer possible.

0d array printing changed, `style` arg of array2string deprecated

Previously the `str` and `repr` of 0d arrays had idiosyncratic implementations which returned `str(a.item())` and `'array(' + repr(a.item()) + ')'` respectively for 0d array `a`, unlike both numpy scalars and higher dimension ndarrays.

Now, the `str` of a 0d array acts like a numpy scalar using `str(a[()])` and the `repr` acts like higher dimension arrays using `formatter(a[()])`, where `formatter` can be specified using `np.set_printoptions`. The `style` argument of `np.array2string` is deprecated.

This new behavior is disabled in 1.13 legacy printing mode, see compatibility notes above.

Seeding `RandomState` using an array requires a 1-d array

`RandomState` previously would accept empty arrays or arrays with 2 or more dimensions, which resulted in either a failure to seed (empty arrays) or for some of the passed values to be ignored when setting the seed.

`MaskedArray` objects show a more useful `repr`

The `repr` of a `MaskedArray` is now closer to the python code that would produce it, with arrays now being shown with commas and dtypes. Like the other formatting changes, this can be disabled with the 1.13 legacy printing mode in order to help transition doctests.

The `repr` of `np.polynomial` classes is more explicit

It now shows the domain and window parameters as keyword arguments to make them more clear:

```
>>> np.polynomial.Polynomial(range(4))
Polynomial([0.,  1.,  2.,  3.], domain=[-1,  1], window=[-1,  1])
```

15.55 NumPy 1.13.3 Release Notes

This is a bugfix release for some problems found since 1.13.1. The most important fixes are for CVE-2017-12852 and temporary elision. Users of earlier versions of 1.13 should upgrade.

The Python versions supported are 2.7 and 3.4 - 3.6. The Python 3.6 wheels available from PIP are built with Python 3.6.2 and should be compatible with all previous versions of Python 3.6. It was cythonized with Cython 0.26.1, which should be free of the bugs found in 0.27 while also being compatible with Python 3.7-dev. The Windows wheels were built with OpenBlas instead ATLAS, which should improve the performance of the linear algebra functions.

The NumPy 1.13.3 release is a re-release of 1.13.2, which suffered from a bug in Cython 0.27.0.

15.55.1 Contributors

A total of 12 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Allan Haldane
- Brandon Carter
- Charles Harris
- Eric Wieser
- Iryna Shcherbina +
- James Bourbeau +
- Jonathan Helmus
- Julian Taylor
- Matti Picus
- Michael Lamparski +
- Michael Seifert
- Ralf Gommers

15.55.2 Pull requests merged

A total of 22 pull requests were merged for this release.

- #9390 BUG: Return the poly1d coefficients array directly
- #9555 BUG: Fix regression in 1.13.x in distutils.mingw32ccompiler.
- #9556 BUG: Fix true_divide when dtype=np.float64 specified.
- #9557 DOC: Fix some rst markup in numpy/doc/basics.py.
- #9558 BLD: Remove -xhost flag from IntelFCompiler.
- #9559 DOC: Removes broken docstring example (source code, png, pdf)…
- #9580 BUG: Add hypot and cabs functions to WIN32 blacklist.
- #9732 BUG: Make scalar function elision check if temp is writeable.
- #9736 BUG: Various fixes to np.gradient
- #9742 BUG: Fix np.pad for CVE-2017-12852
- #9744 BUG: Check for exception in sort functions, add tests
- #9745 DOC: Add whitespace after “versionadded::” directive so it actually…
- #9746 BUG: Memory leak in np.dot of size 0
- #9747 BUG: Adjust gfortran version search regex
- #9757 BUG: Cython 0.27 breaks NumPy on Python 3.
- #9764 BUG: Ensure *_npy_scaled_cexp{,f,l}* is defined when needed.
- #9765 BUG: PyArray_CountNonzero does not check for exceptions
- #9766 BUG: Fixes histogram monotonicity check for unsigned bin values
- #9767 BUG: Ensure consistent result dtype of count_nonzero
- #9771 BUG: MAINT: Fix mtrand for Cython 0.27.
- #9772 DOC: Create the 1.13.2 release notes.
- #9794 DOC: Create 1.13.3 release notes.

15.56 NumPy 1.13.2 Release Notes

This is a bugfix release for some problems found since 1.13.1. The most important fixes are for CVE-2017-12852 and temporary elision. Users of earlier versions of 1.13 should upgrade.

The Python versions supported are 2.7 and 3.4 - 3.6. The Python 3.6 wheels available from PIP are built with Python 3.6.2 and should be compatible with all previous versions of Python 3.6. The Windows wheels are now built with OpenBlas instead ATLAS, which should improve the performance of the linear algebra functions.

15.56.1 Contributors

A total of 12 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Allan Haldane
- Brandon Carter
- Charles Harris
- Eric Wieser
- Iryna Shcherbina +
- James Bourbeau +
- Jonathan Helmus
- Julian Taylor
- Matti Picus
- Michael Lamparski +
- Michael Seifert
- Ralf Gommers

15.56.2 Pull requests merged

A total of 20 pull requests were merged for this release.

- #9390 BUG: Return the poly1d coefficients array directly
- #9555 BUG: Fix regression in 1.13.x in distutils.mingw32ccompiler.
- #9556 BUG: Fix true_divide when dtype=np.float64 specified.
- #9557 DOC: Fix some rst markup in numpy/doc/basics.py.
- #9558 BLD: Remove -xhost flag from IntelFCompiler.
- #9559 DOC: Removes broken docstring example (source code, png, pdf)…
- #9580 BUG: Add hypot and cabs functions to WIN32 blacklist.
- #9732 BUG: Make scalar function elision check if temp is writeable.
- #9736 BUG: Various fixes to np.gradient
- #9742 BUG: Fix np.pad for CVE-2017-12852
- #9744 BUG: Check for exception in sort functions, add tests
- #9745 DOC: Add whitespace after "versionadded::" directive so it actually…
- #9746 BUG: Memory leak in np.dot of size 0
- #9747 BUG: Adjust gfortran version search regex
- #9757 BUG: Cython 0.27 breaks NumPy on Python 3.
- #9764 BUG: Ensure *_npy_scaled_cexp{,f,l}* is defined when needed.
- #9765 BUG: PyArray_CountNonzero does not check for exceptions
- #9766 BUG: Fixes histogram monotonicity check for unsigned bin values

- #9767 BUG: Ensure consistent result dtype of count_nonzero
- #9771 BUG, MAINT: Fix mtrand for Cython 0.27.

15.57 NumPy 1.13.1 Release Notes

This is a bugfix release for problems found in 1.13.0. The major changes are fixes for the new memory overlap detection and temporary elision as well as reversion of the removal of the boolean binary − operator. Users of 1.13.0 should upgrade.

Thr Python versions supported are 2.7 and 3.4 - 3.6. Note that the Python 3.6 wheels available from PIP are built against 3.6.1, hence will not work when used with 3.6.0 due to Python bug 29943. NumPy 1.13.2 will be released shortly after Python 3.6.2 is out to fix that problem. If you are using 3.6.0 the workaround is to upgrade to 3.6.1 or use an earlier Python version.

15.57.1 Pull requests merged

A total of 19 pull requests were merged for this release.

- #9240 DOC: BLD: fix lots of Sphinx warnings/errors.
- #9255 Revert “DEP: Raise TypeError for subtract(bool, bool).”
- #9261 BUG: don’t elide into readonly and updateifcopy temporaries for…
- #9262 BUG: fix missing keyword rename for common block in numpy.f2py
- #9263 BUG: handle resize of 0d array
- #9267 DOC: update f2py front page and some doc build metadata.
- #9299 BUG: Fix Intel compilation on Unix.
- #9317 BUG: fix wrong ndim used in empty where check
- #9319 BUG: Make extensions compilable with MinGW on Py2.7
- #9339 BUG: Prevent crash if ufunc doc string is null
- #9340 BUG: umath: un-break ufunc where= when no out= is given
- #9371 DOC: Add isnat/positive ufunc to documentation
- #9372 BUG: Fix error in fromstring function from numpy.core.records…
- #9373 BUG: ‘)’ is printed at the end pointer of the buffer in numpy.f2py.
- #9374 DOC: Create NumPy 1.13.1 release notes.
- #9376 BUG: Prevent hang traversing ufunc userloop linked list
- #9377 DOC: Use x1 and x2 in the heaviside docstring.
- #9378 DOC: Add $PARAMS to the isnat docstring
- #9379 DOC: Update the 1.13.1 release notes

15.57.2 Contributors

A total of 12 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Andras Deak +
- Bob Eldering +
- Charles Harris
- Daniel Hrisca +
- Eric Wieser
- Joshua Leahy +
- Julian Taylor
- Michael Seifert
- Pauli Virtanen
- Ralf Gommers
- Roland Kaufmann
- Warren Weckesser

15.58 NumPy 1.13.0 Release Notes

This release supports Python 2.7 and 3.4 - 3.6.

15.58.1 Highlights

- Operations like `a + b + c` will reuse temporaries on some platforms, resulting in less memory use and faster execution.
- Inplace operations check if inputs overlap outputs and create temporaries to avoid problems.
- New `__array_ufunc__` attribute provides improved ability for classes to override default ufunc behavior.
- New `np.block` function for creating blocked arrays.

15.58.2 New functions

- New `np.positive` ufunc.
- New `np.divmod` ufunc provides more efficient divmod.
- New `np.isnat` ufunc tests for NaT special values.
- New `np.heaviside` ufunc computes the Heaviside function.
- New `np.isin` function, improves on `in1d`.
- New `np.block` function for creating blocked arrays.
- New `PyArray_MapIterArrayCopyIfOverlap` added to NumPy C-API.

See below for details.

15.58.3 Deprecations

- Calling `np.fix`, `np.isposinf`, and `np.isneginf` with `f(x, y=out)` is deprecated - the argument should be passed as `f(x, out=out)`, which matches other ufunc-like interfaces.
- Use of the C-API `NPY_CHAR` type number deprecated since version 1.7 will now raise deprecation warnings at runtime. Extensions built with older f2py versions need to be recompiled to remove the warning.
- `np.ma.argsort`, `np.ma.minimum.reduce`, and `np.ma.maximum.reduce` should be called with an explicit *axis* argument when applied to arrays with more than 2 dimensions, as the default value of this argument (`None`) is inconsistent with the rest of numpy (`-1`, `0`, and `0`, respectively).
- `np.ma.MaskedArray.mini` is deprecated, as it almost duplicates the functionality of `np.MaskedArray.min`. Exactly equivalent behaviour can be obtained with `np.ma.minimum.reduce`.
- The single-argument form of `np.ma.minimum` and `np.ma.maximum` is deprecated. `np.maximum.` `np.ma.minimum(x)` should now be spelt `np.ma.minimum.reduce(x)`, which is consistent with how this would be done with `np.minimum`.
- Calling `ndarray.conjugate` on non-numeric dtypes is deprecated (it should match the behavior of `np.conjugate`, which throws an error).
- Calling `expand_dims` when the `axis` keyword does not satisfy `-a.ndim - 1 <= axis <= a.ndim`, where `a` is the array being reshaped, is deprecated.

15.58.4 Future Changes

- Assignment between structured arrays with different field names will change in NumPy 1.14. Previously, fields in the dst would be set to the value of the identically-named field in the src. In numpy 1.14 fields will instead be assigned 'by position': The n-th field of the dst will be set to the n-th field of the src array. Note that the `FutureWarning` raised in NumPy 1.12 incorrectly reported this change as scheduled for NumPy 1.13 rather than NumPy 1.14.

15.58.5 Build System Changes

- `numpy.distutils` now automatically determines C-file dependencies with GCC compatible compilers.

15.58.6 Compatibility notes

Error type changes

- `numpy.hstack()` now throws `ValueError` instead of `IndexError` when input is empty.
- Functions taking an axis argument, when that argument is out of range, now throw `np.AxisError` instead of a mixture of `IndexError` and `ValueError`. For backwards compatibility, `AxisError` subclasses both of these.

Tuple object dtypes

Support has been removed for certain obscure dtypes that were unintentionally allowed, of the form `(old_dtype, new_dtype)`, where either of the dtypes is or contains the `object` dtype. As an exception, dtypes of the form `(object, [('name', object)])` are still supported due to evidence of existing use.

DeprecationWarning to error

See Changes section for more detail.

- `partition`, TypeError when non-integer partition index is used.
- `NpyIter_AdvancedNew`, ValueError when `oa_ndim == 0` and `op_axes` is NULL
- `negative(bool_)`, TypeError when negative applied to booleans.
- `subtract(bool_, bool_)`, TypeError when subtracting boolean from boolean.
- `np.equal, np.not_equal`, object identity doesn't override failed comparison.
- `np.equal, np.not_equal`, object identity doesn't override non-boolean comparison.
- Deprecated boolean indexing behavior dropped. See Changes below for details.
- Deprecated `np.alterdot()` and `np.restoredot()` removed.

FutureWarning to changed behavior

See Changes section for more detail.

- `numpy.average` preserves subclasses
- `array == None` and `array != None` do element-wise comparison.
- `np.equal, np.not_equal`, object identity doesn't override comparison result.

dtypes are now always true

Previously `bool(dtype)` would fall back to the default python implementation, which checked if `len(dtype) > 0`. Since `dtype` objects implement `__len__` as the number of record fields, `bool` of scalar dtypes would evaluate to `False`, which was unintuitive. Now `bool(dtype) == True` for all dtypes.

`__getslice__` and `__setslice__` are no longer needed in `ndarray` subclasses

When subclassing np.ndarray in Python 2.7, it is no longer _necessary_ to implement `__*slice__` on the derived class, as `__*item__` will intercept these calls correctly.

Any code that did implement these will work exactly as before. Code that invokes"ndarray.__getslice__" (e.g. through `super(...).__getslice__`) will now issue a DeprecationWarning - `.__getitem__(slice(start, end))` should be used instead.

Indexing MaskedArrays/Constants with `...` (ellipsis) now returns MaskedArray

This behavior mirrors that of np.ndarray, and accounts for nested arrays in MaskedArrays of object dtype, and ellipsis combined with other forms of indexing.

15.58.7 C API changes

GUfuncs on empty arrays and NpyIter axis removal

It is now allowed to remove a zero-sized axis from NpyIter. Which may mean that code removing axes from NpyIter has to add an additional check when accessing the removed dimensions later on.

The largest followup change is that gufuncs are now allowed to have zero-sized inner dimensions. This means that a gufunc now has to anticipate an empty inner dimension, while this was never possible and an error raised instead.

For most gufuncs no change should be necessary. However, it is now possible for gufuncs with a signature such as `(..., N, M) -> (..., M)` to return a valid result if `N=0` without further wrapping code.

`PyArray_MapIterArrayCopyIfOverlap` added to NumPy C-API

Similar to `PyArray_MapIterArray` but with an additional `copy_if_overlap` argument. If `copy_if_overlap != 0`, checks if input has memory overlap with any of the other arrays and make copies as appropriate to avoid problems if the input is modified during the iteration. See the documentation for more complete documentation.

15.58.8 New Features

`__array_ufunc__` added

This is the renamed and redesigned `__numpy_ufunc__`. Any class, ndarray subclass or not, can define this method or set it to `None` in order to override the behavior of NumPy's ufuncs. This works quite similarly to Python's `__mul__` and other binary operation routines. See the documentation for a more detailed description of the implementation and behavior of this new option. The API is provisional, we do not yet guarantee backward compatibility as modifications may be made pending feedback. See NEP 13 and documentation for more details.

New `positive` ufunc

This ufunc corresponds to unary +, but unlike + on an ndarray it will raise an error if array values do not support numeric operations.

New `divmod` ufunc

This ufunc corresponds to the Python builtin *divmod*, and is used to implement *divmod* when called on numpy arrays. `np.divmod(x, y)` calculates a result equivalent to `(np.floor_divide(x, y), np.remainder(x, y))` but is approximately twice as fast as calling the functions separately.

`np.isnat` ufunc tests for NaT special datetime and timedelta values

The new ufunc `np.isnat` finds the positions of special NaT values within datetime and timedelta arrays. This is analogous to `np.isnan`.

`np.heaviside` ufunc computes the Heaviside function

The new function `np.heaviside(x, h0)` (a ufunc) computes the Heaviside function:

```
                    { 0   if x < 0,
heaviside(x, h0) = { h0  if x == 0,
                    { 1   if x > 0.
```

`np.block` function for creating blocked arrays

Add a new `block` function to the current stacking functions `vstack`, `hstack`, and `stack`. This allows concatenation across multiple axes simultaneously, with a similar syntax to array creation, but where elements can themselves be arrays. For instance:

```
>>> A = np.eye(2) * 2
>>> B = np.eye(3) * 3
>>> np.block([
...     [A,               np.zeros((2, 3))],
...     [np.ones((3, 2)), B               ]
... ])
array([[ 2.,  0.,  0.,  0.,  0.],
       [ 0.,  2.,  0.,  0.,  0.],
       [ 1.,  1.,  3.,  0.,  0.],
       [ 1.,  1.,  0.,  3.,  0.],
       [ 1.,  1.,  0.,  0.,  3.]])
```

While primarily useful for block matrices, this works for arbitrary dimensions of arrays.

It is similar to Matlab's square bracket notation for creating block matrices.

`isin` function, improving on `in1d`

The new function `isin` tests whether each element of an N-dimensional array is present anywhere within a second array. It is an enhancement of `in1d` that preserves the shape of the first array.

Temporary elision

On platforms providing the `backtrace` function NumPy will try to avoid creating temporaries in expression involving basic numeric types. For example `d = a + b + c` is transformed to `d = a + b; d += c` which can improve performance for large arrays as less memory bandwidth is required to perform the operation.

`axes` argument for `unique`

In an N-dimensional array, the user can now choose the axis along which to look for duplicate N-1-dimensional elements using `numpy.unique`. The original behaviour is recovered if `axis=None` (default).

`np.gradient` now supports unevenly spaced data

Users can now specify a not-constant spacing for data. In particular `np.gradient` can now take:

1. A single scalar to specify a sample distance for all dimensions.
2. N scalars to specify a constant sample distance for each dimension. i.e. `dx`, `dy`, `dz`, …
3. N arrays to specify the coordinates of the values along each dimension of F. The length of the array must match the size of the corresponding dimension
4. Any combination of N scalars/arrays with the meaning of 2. and 3.

This means that, e.g., it is now possible to do the following:

```
>>> f = np.array([[1, 2, 6], [3, 4, 5]], dtype=np.float_)
>>> dx = 2.
>>> y = [1., 1.5, 3.5]
>>> np.gradient(f, dx, y)
[array([[ 1. ,  1. , -0.5], [ 1. ,  1. , -0.5]]),
 array([[ 2. ,  2. ,  2. ], [ 2. ,  1.7,  0.5]])]
```

Support for returning arrays of arbitrary dimensions in `apply_along_axis`

Previously, only scalars or 1D arrays could be returned by the function passed to `apply_along_axis`. Now, it can return an array of any dimensionality (including 0D), and the shape of this array replaces the axis of the array being iterated over.

`.ndim` property added to `dtype` to complement `.shape`

For consistency with `ndarray` and `broadcast`, `d.ndim` is a shorthand for `len(d.shape)`.

Support for tracemalloc in Python 3.6

NumPy now supports memory tracing with tracemalloc module of Python 3.6 or newer. Memory allocations from NumPy are placed into the domain defined by `numpy.lib.tracemalloc_domain`. Note that NumPy allocation will not show up in tracemalloc of earlier Python versions.

NumPy may be built with relaxed stride checking debugging

Setting NPY_RELAXED_STRIDES_DEBUG=1 in the environment when relaxed stride checking is enabled will cause NumPy to be compiled with the affected strides set to the maximum value of npy_intp in order to help detect invalid usage of the strides in downstream projects. When enabled, invalid usage often results in an error being raised, but the exact type of error depends on the details of the code. TypeError and OverflowError have been observed in the wild.

It was previously the case that this option was disabled for releases and enabled in master and changing between the two required editing the code. It is now disabled by default but can be enabled for test builds.

15.58.9 Improvements

Ufunc behavior for overlapping inputs

Operations where ufunc input and output operands have memory overlap produced undefined results in previous NumPy versions, due to data dependency issues. In NumPy 1.13.0, results from such operations are now defined to be the same as for equivalent operations where there is no memory overlap.

Operations affected now make temporary copies, as needed to eliminate data dependency. As detecting these cases is computationally expensive, a heuristic is used, which may in rare cases result to needless temporary copies. For operations where the data dependency is simple enough for the heuristic to analyze, temporary copies will not be made even if the arrays overlap, if it can be deduced copies are not necessary. As an example,"np.add(a, b, out=a)" will not involve copies.

To illustrate a previously undefined operation:

```
>>> x = np.arange(16).astype(float)
>>> np.add(x[1:], x[:-1], out=x[1:])
```

In NumPy 1.13.0 the last line is guaranteed to be equivalent to:

```
>>> np.add(x[1:].copy(), x[:-1].copy(), out=x[1:])
```

A similar operation with simple non-problematic data dependence is:

```
>>> x = np.arange(16).astype(float)
>>> np.add(x[1:], x[:-1], out=x[:-1])
```

It will continue to produce the same results as in previous NumPy versions, and will not involve unnecessary temporary copies.

The change applies also to in-place binary operations, for example:

```
>>> x = np.random.rand(500, 500)
>>> x += x.T
```

This statement is now guaranteed to be equivalent to `x[...] = x + x.T`, whereas in previous NumPy versions the results were undefined.

Partial support for 64-bit f2py extensions with MinGW

Extensions that incorporate Fortran libraries can now be built using the free MinGW toolset, also under Python 3.5. This works best for extensions that only do calculations and uses the runtime modestly (reading and writing from files, for instance). Note that this does not remove the need for Mingwpy; if you make extensive use of the runtime, you will most likely run into issues. Instead, it should be regarded as a band-aid until Mingwpy is fully functional.

Extensions can also be compiled using the MinGW toolset using the runtime library from the (moveable) WinPython 3.4 distribution, which can be useful for programs with a PySide1/Qt4 front-end.

Performance improvements for `packbits` and `unpackbits`

The functions `numpy.packbits` with boolean input and `numpy.unpackbits` have been optimized to be a significantly faster for contiguous data.

Fix for PPC long double floating point information

In previous versions of NumPy, the `finfo` function returned invalid information about the double double format of the `longdouble` float type on Power PC (PPC). The invalid values resulted from the failure of the NumPy algorithm to deal with the variable number of digits in the significand that are a feature of *PPC long doubles*. This release by-passes the failing algorithm by using heuristics to detect the presence of the PPC double double format. A side-effect of using these heuristics is that the `finfo` function is faster than previous releases.

Better default repr for `ndarray` subclasses

Subclasses of ndarray with no `repr` specialization now correctly indent their data and type lines.

More reliable comparisons of masked arrays

Comparisons of masked arrays were buggy for masked scalars and failed for structured arrays with dimension higher than one. Both problems are now solved. In the process, it was ensured that in getting the result for a structured array, masked fields are properly ignored, i.e., the result is equal if all fields that are non-masked in both are equal, thus making the behaviour identical to what one gets by comparing an unstructured masked array and then doing `.all()` over some axis.

np.matrix with booleans elements can now be created using the string syntax

`np.matrix` failed whenever one attempts to use it with booleans, e.g., `np.matrix('True')`. Now, this works as expected.

More `linalg` operations now accept empty vectors and matrices

All of the following functions in `np.linalg` now work when given input arrays with a 0 in the last two dimensions: `det`, `slogdet`, `pinv`, `eigvals`, `eigvalsh`, `eig`, `eigh`.

Bundled version of LAPACK is now 3.2.2

NumPy comes bundled with a minimal implementation of lapack for systems without a lapack library installed, under the name of `lapack_lite`. This has been upgraded from LAPACK 3.0.0 (June 30, 1999) to LAPACK 3.2.2 (June 30, 2010). See the LAPACK changelogs for details on the all the changes this entails.

While no new features are exposed through `numpy`, this fixes some bugs regarding "workspace" sizes, and in some places may use faster algorithms.

`reduce` of `np.hypot.reduce` and `np.logical_xor` allowed in more cases

This now works on empty arrays, returning 0, and can reduce over multiple axes. Previously, a `ValueError` was thrown in these cases.

Better `repr` of object arrays

Object arrays that contain themselves no longer cause a recursion error.

Object arrays that contain `list` objects are now printed in a way that makes clear the difference between a 2d object array, and a 1d object array of lists.

15.58.10 Changes

`argsort` on masked arrays takes the same default arguments as `sort`

By default, `argsort` now places the masked values at the end of the sorted array, in the same way that `sort` already did. Additionally, the `end_with` argument is added to `argsort`, for consistency with `sort`. Note that this argument is not added at the end, so breaks any code that passed `fill_value` as a positional argument.

`average` now preserves subclasses

For ndarray subclasses, `numpy.average` will now return an instance of the subclass, matching the behavior of most other NumPy functions such as `mean`. As a consequence, also calls that returned a scalar may now return a subclass array scalar.

`array == None` and `array != None` do element-wise comparison

Previously these operations returned scalars `False` and `True` respectively.

`np.equal, np.not_equal` for object arrays ignores object identity

Previously, these functions always treated identical objects as equal. This had the effect of overriding comparison failures, comparison of objects that did not return booleans, such as np.arrays, and comparison of objects where the results differed from object identity, such as NaNs.

Boolean indexing changes

- Boolean array-likes (such as lists of python bools) are always treated as boolean indexes.
- Boolean scalars (including python `True`) are legal boolean indexes and never treated as integers.
- Boolean indexes must match the dimension of the axis that they index.
- Boolean indexes used on the lhs of an assignment must match the dimensions of the rhs.
- Boolean indexing into scalar arrays return a new 1-d array. This means that `array(1)[array(True)]` gives `array([1])` and not the original array.

`np.random.multivariate_normal` behavior with bad covariance matrix

It is now possible to adjust the behavior the function will have when dealing with the covariance matrix by using two new keyword arguments:

- `tol` can be used to specify a tolerance to use when checking that the covariance matrix is positive semidefinite.
- `check_valid` can be used to configure what the function will do in the presence of a matrix that is not positive semidefinite. Valid options are `ignore`, `warn` and `raise`. The default value, `warn` keeps the the behavior used on previous releases.

`assert_array_less` compares `np.inf` and `-np.inf` now

Previously, `np.testing.assert_array_less` ignored all infinite values. This is not the expected behavior both according to documentation and intuitively. Now, -inf < x < inf is considered `True` for any real number x and all other cases fail.

`assert_array_` and masked arrays `assert_equal` hide less warnings

Some warnings that were previously hidden by the `assert_array_` functions are not hidden anymore. In most cases the warnings should be correct and, should they occur, will require changes to the tests using these functions. For the masked array `assert_equal` version, warnings may occur when comparing NaT. The function presently does not handle NaT or NaN specifically and it may be best to avoid it at this time should a warning show up due to this change.

`offset` attribute value in `memmap` objects

The `offset` attribute in a `memmap` object is now set to the offset into the file. This is a behaviour change only for offsets greater than `mmap.ALLOCATIONGRANULARITY`.

`np.real` and `np.imag` return scalars for scalar inputs

Previously, `np.real` and `np.imag` used to return array objects when provided a scalar input, which was inconsistent with other functions like `np.angle` and `np.conj`.

The polynomial convenience classes cannot be passed to ufuncs

The ABCPolyBase class, from which the convenience classes are derived, sets `__array_ufun__ = None` in order of opt out of ufuncs. If a polynomial convenience class instance is passed as an argument to a ufunc, a `TypeError` will now be raised.

Output arguments to ufuncs can be tuples also for ufunc methods

For calls to ufuncs, it was already possible, and recommended, to use an `out` argument with a tuple for ufuncs with multiple outputs. This has now been extended to output arguments in the `reduce`, `accumulate`, and `reduceat` methods. This is mostly for compatibility with `__array_ufunc`; there are no ufuncs yet that have more than one output.

15.59 NumPy 1.12.1 Release Notes

NumPy 1.12.1 supports Python 2.7 and 3.4 - 3.6 and fixes bugs and regressions found in NumPy 1.12.0. In particular, the regression in f2py constant parsing is fixed. Wheels for Linux, Windows, and OSX can be found on PyPI,

15.59.1 Bugs Fixed

- BUG: Fix wrong future nat warning and equiv type logic error…
- BUG: Fix wrong masked median for some special cases
- DOC: Place np.average in inline code
- TST: Work around isfinite inconsistency on i386
- BUG: Guard against replacing constants without '_' spec in f2py.
- BUG: Fix mean for float 16 non-array inputs for 1.12
- BUG: Fix calling python api with error set and minor leaks for…
- BUG: Make iscomplexobj compatible with custom dtypes again
- BUG: Fix undefined behaviour induced by bad __array_wrap__
- BUG: Fix MaskedArray.__setitem__
- BUG: PPC64el machines are POWER for Fortran in f2py
- BUG: Look up methods on MaskedArray in *_frommethod*
- BUG: Remove extra digit in binary_repr at limit
- BUG: Fix deepcopy regression for empty arrays.
- BUG: Fix ma.median for empty ndarrays

15.60 NumPy 1.12.0 Release Notes

This release supports Python 2.7 and 3.4 - 3.6.

15.60.1 Highlights

The NumPy 1.12.0 release contains a large number of fixes and improvements, but few that stand out above all others. That makes picking out the highlights somewhat arbitrary but the following may be of particular interest or indicate areas likely to have future consequences.

- Order of operations in `np.einsum` can now be optimized for large speed improvements.
- New `signature` argument to `np.vectorize` for vectorizing with core dimensions.
- The `keepdims` argument was added to many functions.
- New context manager for testing warnings
- Support for BLIS in numpy.distutils
- Much improved support for PyPy (not yet finished)

15.60.2 Dropped Support

- Support for Python 2.6, 3.2, and 3.3 has been dropped.

15.60.3 Added Support

- Support for PyPy 2.7 v5.6.0 has been added. While not complete (nditer `updateifcopy` is not supported yet), this is a milestone for PyPy's C-API compatibility layer.

15.60.4 Build System Changes

- Library order is preserved, instead of being reordered to match that of the directories.

15.60.5 Deprecations

Assignment of ndarray object's `data` attribute

Assigning the 'data' attribute is an inherently unsafe operation as pointed out in gh-7083. Such a capability will be removed in the future.

Unsafe int casting of the num attribute in `linspace`

`np.linspace` now raises DeprecationWarning when num cannot be safely interpreted as an integer.

Insufficient bit width parameter to `binary_repr`

If a 'width' parameter is passed into `binary_repr` that is insufficient to represent the number in base 2 (positive) or 2's complement (negative) form, the function used to silently ignore the parameter and return a representation using the minimal number of bits needed for the form in question. Such behavior is now considered unsafe from a user perspective and will raise an error in the future.

15.60.6 Future Changes

- In 1.13 NAT will always compare False except for `NAT != NAT`, which will be True. In short, NAT will behave like NaN
- In 1.13 `np.average` will preserve subclasses, to match the behavior of most other numpy functions such as np.mean. In particular, this means calls which returned a scalar may return a 0-d subclass object instead.

Multiple-field manipulation of structured arrays

In 1.13 the behavior of structured arrays involving multiple fields will change in two ways:

First, indexing a structured array with multiple fields (eg, `arr[['f1', 'f3']]`) will return a view into the original array in 1.13, instead of a copy. Note the returned view will have extra padding bytes corresponding to intervening fields in the original array, unlike the copy in 1.12, which will affect code such as `arr[['f1', 'f3']].view(newdtype)`.

Second, for numpy versions 1.6 to 1.12 assignment between structured arrays occurs "by field name": Fields in the destination array are set to the identically-named field in the source array or to 0 if the source does not have a field:

```
>>> a = np.array([(1,2),(3,4)], dtype=[('x', 'i4'), ('y', 'i4')])
>>> b = np.ones(2, dtype=[('z', 'i4'), ('y', 'i4'), ('x', 'i4')])
>>> b[:] = a
>>> b
array([(0, 2, 1), (0, 4, 3)],
      dtype=[('z', '<i4'), ('y', '<i4'), ('x', '<i4')])
```

In 1.13 assignment will instead occur "by position": The Nth field of the destination will be set to the Nth field of the source regardless of field name. The old behavior can be obtained by using indexing to reorder the fields before assignment, e.g., `b[['x', 'y']] = a[['y', 'x']]`.

15.60.7 Compatibility notes

DeprecationWarning to error

- Indexing with floats raises `IndexError`, e.g., a[0, 0.0].
- Indexing with non-integer array_like raises `IndexError`, e.g., `a['1', '2']`
- Indexing with multiple ellipsis raises `IndexError`, e.g., `a[..., ...]`.
- Non-integers used as index values raise `TypeError`, e.g., in `reshape`, `take`, and specifying reduce axis.

FutureWarning to changed behavior

- `np.full` now returns an array of the fill-value's dtype if no dtype is given, instead of defaulting to float.
- `np.average` will emit a warning if the argument is a subclass of ndarray, as the subclass will be preserved starting in 1.13. (see Future Changes)

`power` and `**` raise errors for integer to negative integer powers

The previous behavior depended on whether numpy scalar integers or numpy integer arrays were involved.

For arrays

- Zero to negative integer powers returned least integral value.
- Both 1, -1 to negative integer powers returned correct values.
- The remaining integers returned zero when raised to negative integer powers.

For scalars

- Zero to negative integer powers returned least integral value.
- Both 1, -1 to negative integer powers returned correct values.

- The remaining integers sometimes returned zero, sometimes the correct float depending on the integer type combination.

All of these cases now raise a `ValueError` except for those integer combinations whose common type is float, for instance uint64 and int8. It was felt that a simple rule was the best way to go rather than have special exceptions for the integer units. If you need negative powers, use an inexact type.

Relaxed stride checking is the default

This will have some impact on code that assumed that `F_CONTIGUOUS` and `C_CONTIGUOUS` were mutually exclusive and could be set to determine the default order for arrays that are now both.

The `np.percentile` 'midpoint' interpolation method fixed for exact indices

The 'midpoint' interpolator now gives the same result as 'lower' and 'higher' when the two coincide. Previous behavior of 'lower' + 0.5 is fixed.

`keepdims` kwarg is passed through to user-class methods

numpy functions that take a `keepdims` kwarg now pass the value through to the corresponding methods on ndarray sub-classes. Previously the `keepdims` keyword would be silently dropped. These functions now have the following behavior:

1. If user does not provide `keepdims`, no keyword is passed to the underlying method.
2. Any user-provided value of `keepdims` is passed through as a keyword argument to the method.

This will raise in the case where the method does not support a `keepdims` kwarg and the user explicitly passes in `keepdims`.

The following functions are changed: `sum`, `product`, `sometrue`, `alltrue`, `any`, `all`, `amax`, `amin`, `prod`, `mean`, `std`, `var`, `nanmin`, `nanmax`, `nansum`, `nanprod`, `nanmean`, `nanmedian`, `nanvar`, `nanstd`

`bitwise_and` identity changed

The previous identity was 1, it is now -1. See entry in Improvements for more explanation.

ma.median warns and returns nan when unmasked invalid values are encountered

Similar to unmasked median the masked median *ma.median* now emits a Runtime warning and returns *NaN* in slices where an unmasked *NaN* is present.

Greater consistency in `assert_almost_equal`

The precision check for scalars has been changed to match that for arrays. It is now:

```
abs(actual - desired) < 1.5 * 10**(-decimal)
```

Note that this is looser than previously documented, but agrees with the previous implementation used in `assert_array_almost_equal`. Due to the change in implementation some very delicate tests may fail that did not fail before.

`NoseTester` behaviour of warnings during testing

When `raise_warnings="develop"` is given, all uncaught warnings will now be considered a test failure. Previously only selected ones were raised. Warnings which are not caught or raised (mostly when in release mode) will be shown once during the test cycle similar to the default python settings.

`assert_warns` and `deprecated` decorator more specific

The `assert_warns` function and context manager are now more specific to the given warning category. This increased specificity leads to them being handled according to the outer warning settings. This means that no warning may be raised in cases where a wrong category warning is given and ignored outside the context. Alternatively the increased specificity may mean that warnings that were incorrectly ignored will now be shown or raised. See also the new `suppress_warnings` context manager. The same is true for the `deprecated` decorator.

C API

No changes.

15.60.8 New Features

Writeable keyword argument for `as_strided`

`np.lib.stride_tricks.as_strided` now has a `writeable` keyword argument. It can be set to False when no write operation to the returned array is expected to avoid accidental unpredictable writes.

`axes` keyword argument for `rot90`

The `axes` keyword argument in `rot90` determines the plane in which the array is rotated. It defaults to `axes=(0,1)` as in the original function.

Generalized `flip`

`flipud` and `fliplr` reverse the elements of an array along axis=0 and axis=1 respectively. The newly added `flip` function reverses the elements of an array along any given axis.

- `np.count_nonzero` now has an `axis` parameter, allowing non-zero counts to be generated on more than just a flattened array object.

BLIS support in `numpy.distutils`

Building against the BLAS implementation provided by the BLIS library is now supported. See the `[blis]` section in `site.cfg.example` (in the root of the numpy repo or source distribution).

Hook in `numpy/__init__.py` to run distribution-specific checks

Binary distributions of numpy may need to run specific hardware checks or load specific libraries during numpy initialization. For example, if we are distributing numpy with a BLAS library that requires SSE2 instructions, we would like to check the machine on which numpy is running does have SSE2 in order to give an informative error.

Add a hook in `numpy/__init__.py` to import a `numpy/_distributor_init.py` file that will remain empty (bar a docstring) in the standard numpy source, but that can be overwritten by people making binary distributions of numpy.

New nanfunctions `nancumsum` and `nancumprod` added

Nan-functions `nancumsum` and `nancumprod` have been added to compute `cumsum` and `cumprod` by ignoring nans.

`np.interp` can now interpolate complex values

`np.lib.interp(x, xp, fp)` now allows the interpolated array `fp` to be complex and will interpolate at `complex128` precision.

New polynomial evaluation function `polyvalfromroots` added

The new function `polyvalfromroots` evaluates a polynomial at given points from the roots of the polynomial. This is useful for higher order polynomials, where expansion into polynomial coefficients is inaccurate at machine precision.

New array creation function `geomspace` added

The new function `geomspace` generates a geometric sequence. It is similar to `logspace`, but with start and stop specified directly: `geomspace(start, stop)` behaves the same as `logspace(log10(start), log10(stop))`.

New context manager for testing warnings

A new context manager `suppress_warnings` has been added to the testing utils. This context manager is designed to help reliably test warnings. Specifically to reliably filter/ignore warnings. Ignoring warnings by using an "ignore" filter in Python versions before 3.4.x can quickly result in these (or similar) warnings not being tested reliably.

The context manager allows to filter (as well as record) warnings similar to the `catch_warnings` context, but allows for easier specificity. Also printing warnings that have not been filtered or nesting the context manager will work as expected. Additionally, it is possible to use the context manager as a decorator which can be useful when multiple tests give need to hide the same warning.

New masked array functions `ma.convolve` and `ma.correlate` added

These functions wrapped the non-masked versions, but propagate through masked values. There are two different propagation modes. The default causes masked values to contaminate the result with masks, but the other mode only outputs masks if there is no alternative.

New `float_power` ufunc

The new `float_power` ufunc is like the `power` function except all computation is done in a minimum precision of float64. There was a long discussion on the numpy mailing list of how to treat integers to negative integer powers and a popular proposal was that the `__pow__` operator should always return results of at least float64 precision. The `float_power` function implements that option. Note that it does not support object arrays.

`np.loadtxt` now supports a single integer as `usecol` argument

Instead of using `usecol=(n,)` to read the nth column of a file it is now allowed to use `usecol=n`. Also the error message is more user friendly when a non-integer is passed as a column index.

Improved automated bin estimators for `histogram`

Added 'doane' and 'sqrt' estimators to `histogram` via the `bins` argument. Added support for range-restricted histograms with automated bin estimation.

`np.roll` can now roll multiple axes at the same time

The `shift` and `axis` arguments to `roll` are now broadcast against each other, and each specified axis is shifted accordingly.

The `__complex__` method has been implemented for the ndarrays

Calling `complex()` on a size 1 array will now cast to a python complex.

`pathlib.Path` objects now supported

The standard `np.load`, `np.save`, `np.loadtxt`, `np.savez`, and similar functions can now take `pathlib.Path` objects as an argument instead of a filename or open file object.

New `bits` attribute for `np.finfo`

This makes `np.finfo` consistent with `np.iinfo` which already has that attribute.

New `signature` argument to `np.vectorize`

This argument allows for vectorizing user defined functions with core dimensions, in the style of NumPy's generalized universal functions. This allows for vectorizing a much broader class of functions. For example, an arbitrary distance metric that combines two vectors to produce a scalar could be vectorized with `signature='(n),(n)->()'`. See `np.vectorize` for full details.

Emit py3kwarnings for division of integer arrays

To help people migrate their code bases from Python 2 to Python 3, the python interpreter has a handy option -3, which issues warnings at runtime. One of its warnings is for integer division:

```
$ python -3 -c "2/3"

-c:1: DeprecationWarning: classic int division
```

In Python 3, the new integer division semantics also apply to numpy arrays. With this version, numpy will emit a similar warning:

```
$ python -3 -c "import numpy as np; np.array(2)/np.array(3)"

-c:1: DeprecationWarning: numpy: classic int division
```

Previously, it included str (bytes) and unicode on Python2, but only str (unicode) on Python3.

15.60.9 Improvements

`bitwise_and` identity changed

The previous identity was 1 with the result that all bits except the LSB were masked out when the reduce method was used. The new identity is -1, which should work properly on twos complement machines as all bits will be set to one.

Generalized Ufuncs will now unlock the GIL

Generalized Ufuncs, including most of the linalg module, will now unlock the Python global interpreter lock.

Caches in *np.fft* are now bounded in total size and item count

The caches in *np.fft* that speed up successive FFTs of the same length can no longer grow without bounds. They have been replaced with LRU (least recently used) caches that automatically evict no longer needed items if either the memory size or item count limit has been reached.

Improved handling of zero-width string/unicode dtypes

Fixed several interfaces that explicitly disallowed arrays with zero-width string dtypes (i.e. `dtype('S0')` or `dtype('U0')`, and fixed several bugs where such dtypes were not handled properly. In particular, changed `ndarray.__new__` to not implicitly convert `dtype('S0')` to `dtype('S1')` (and likewise for unicode) when creating new arrays.

Integer ufuncs vectorized with AVX2

If the cpu supports it at runtime the basic integer ufuncs now use AVX2 instructions. This feature is currently only available when compiled with GCC.

Order of operations optimization in `np.einsum`

`np.einsum` now supports the `optimize` argument which will optimize the order of contraction. For example, `np.einsum` would complete the chain dot example `np.einsum('ij,jk,kl->il', a, b, c)` in a single pass which would scale like `N^4`; however, when `optimize=True` `np.einsum` will create an intermediate array to reduce this scaling to `N^3` or effectively `np.dot(a, b).dot(c)`. Usage of intermediate tensors to reduce scaling has been applied to the general einsum summation notation. See `np.einsum_path` for more details.

quicksort has been changed to an introsort

The quicksort kind of `np.sort` and `np.argsort` is now an introsort which is regular quicksort but changing to a heapsort when not enough progress is made. This retains the good quicksort performance while changing the worst case runtime from `O(N^2)` to `O(N*log(N))`.

`ediff1d` improved performance and subclass handling

The ediff1d function uses an array instead on a flat iterator for the subtraction. When to_begin or to_end is not None, the subtraction is performed in place to eliminate a copy operation. A side effect is that certain subclasses are handled better, namely astropy.Quantity, since the complete array is created, wrapped, and then begin and end values are set, instead of using concatenate.

Improved precision of `ndarray.mean` for float16 arrays

The computation of the mean of float16 arrays is now carried out in float32 for improved precision. This should be useful in packages such as Theano where the precision of float16 is adequate and its smaller footprint is desirable.

15.60.10 Changes

All array-like methods are now called with keyword arguments in fromnumeric.py

Internally, many array-like methods in fromnumeric.py were being called with positional arguments instead of keyword arguments as their external signatures were doing. This caused a complication in the downstream 'pandas' library that encountered an issue with 'numpy' compatibility. Now, all array-like methods in this module are called with keyword arguments instead.

Operations on np.memmap objects return numpy arrays in most cases

Previously operations on a memmap object would misleadingly return a memmap instance even if the result was actually not memmapped. For example, `arr + 1` or `arr + arr` would return memmap instances, although no memory from the output array is memmapped. Version 1.12 returns ordinary numpy arrays from these operations.

Also, reduction of a memmap (e.g. `.sum(axis=None)`) now returns a numpy scalar instead of a 0d memmap.

stacklevel of warnings increased

The stacklevel for python based warnings was increased so that most warnings will report the offending line of the user code instead of the line the warning itself is given. Passing of stacklevel is now tested to ensure that new warnings will receive the `stacklevel` argument.

This causes warnings with the "default" or "module" filter to be shown once for every offending user code line or user module instead of only once. On python versions before 3.4, this can cause warnings to appear that were falsely ignored before, which may be surprising especially in test suits.

15.61 NumPy 1.11.3 Release Notes

Numpy 1.11.3 fixes a bug that leads to file corruption when very large files opened in append mode are used in `ndarray.tofile`. It supports Python versions 2.6 - 2.7 and 3.2 - 3.5. Wheels for Linux, Windows, and OS X can be found on PyPI.

15.61.1 Contributors to maintenance/1.11.3

A total of 2 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Charles Harris
- Pavel Potocek +

15.61.2 Pull Requests Merged

- #8341: BUG: Fix ndarray.tofile large file corruption in append mode.
- #8346: TST: Fix tests in PR #8341 for NumPy 1.11.x

15.62 NumPy 1.11.2 Release Notes

Numpy 1.11.2 supports Python 2.6 - 2.7 and 3.2 - 3.5. It fixes bugs and regressions found in Numpy 1.11.1 and includes several build related improvements. Wheels for Linux, Windows, and OS X can be found on PyPI.

15.62.1 Pull Requests Merged

Fixes overridden by later merges and release notes updates are omitted.

- #7736 BUG: Many functions silently drop 'keepdims' kwarg.
- #7738 ENH: Add extra kwargs and update doc of many MA methods.
- #7778 DOC: Update Numpy 1.11.1 release notes.
- #7793 BUG: MaskedArray.count treats negative axes incorrectly.
- #7816 BUG: Fix array too big error for wide dtypes.
- #7821 BUG: Make sure npy_mul_with_overflow_<type> detects overflow.
- #7824 MAINT: Allocate fewer bytes for empty arrays.
- #7847 MAINT,DOC: Fix some imp module uses and update f2py.compile docstring.
- #7849 MAINT: Fix remaining uses of deprecated Python imp module.
- #7851 BLD: Fix ATLAS version detection.
- #7896 BUG: Construct ma.array from np.array which contains padding.
- #7904 BUG: Fix float16 type not being called due to wrong ordering.
- #7917 BUG: Production install of numpy should not require nose.
- #7919 BLD: Fixed MKL detection for recent versions of this library.
- #7920 BUG: Fix for issue #7835 (ma.median of 1d).
- #7932 BUG: Monkey-patch _msvccompile.gen_lib_option like other compilers.
- #7939 BUG: Check for HAVE_LDOUBLE_DOUBLE_DOUBLE_LE in npy_math_complex.
- #7953 BUG: Guard against buggy comparisons in generic quicksort.
- #7954 BUG: Use keyword arguments to initialize Extension base class.
- #7955 BUG: Make sure numpy globals keep identity after reload.
- #7972 BUG: MSVCCompiler grows 'lib' & 'include' env strings exponentially.
- #8005 BLD: Remove __NUMPY_SETUP__ from builtins at end of setup.py.
- #8010 MAINT: Remove leftover imp module imports.
- #8020 BUG: Fix return of np.ma.count if keepdims is True and axis is None.
- #8024 BUG: Fix numpy.ma.median.
- #8031 BUG: Fix np.ma.median with only one non-masked value.
- #8044 BUG: Fix bug in NpyIter buffering with discontinuous arrays.

15.63 NumPy 1.11.1 Release Notes

Numpy 1.11.1 supports Python 2.6 - 2.7 and 3.2 - 3.5. It fixes bugs and regressions found in Numpy 1.11.0 and includes several build related improvements. Wheels for Linux, Windows, and OSX can be found on PyPI.

15.63.1 Fixes Merged

- #7506 BUG: Make sure numpy imports on python 2.6 when nose is unavailable.
- #7530 BUG: Floating exception with invalid axis in np.lexsort.
- #7535 BUG: Extend glibc complex trig functions blacklist to glibc < 2.18.
- #7551 BUG: Allow graceful recovery for no compiler.
- #7558 BUG: Constant padding expected wrong type in constant_values.
- #7578 BUG: Fix OverflowError in Python 3.x. in swig interface.
- #7590 BLD: Fix configparser.InterpolationSyntaxError.
- #7597 BUG: Make np.ma.take work on scalars.
- #7608 BUG: linalg.norm(): Don't convert object arrays to float.
- #7638 BLD: Correct C compiler customization in system_info.py.
- #7654 BUG: ma.median of 1d array should return a scalar.
- #7656 BLD: Remove hardcoded Intel compiler flag -xSSE4.2.
- #7660 BUG: Temporary fix for str(mvoid) for object field types.
- #7665 BUG: Fix incorrect printing of 1D masked arrays.
- #7670 BUG: Correct initial index estimate in histogram.
- #7671 BUG: Boolean assignment no GIL release when transfer needs API.
- #7676 BUG: Fix handling of right edge of final histogram bin.
- #7680 BUG: Fix np.clip bug NaN handling for Visual Studio 2015.
- #7724 BUG: Fix segfaults in np.random.shuffle.
- #7731 MAINT: Change mkl_info.dir_env_var from MKL to MKLROOT.
- #7737 BUG: Fix issue on OS X with Python 3.x, npymath.ini not installed.

15.64 NumPy 1.11.0 Release Notes

This release supports Python 2.6 - 2.7 and 3.2 - 3.5 and contains a number of enhancements and improvements. Note also the build system changes listed below as they may have subtle effects.

No Windows (TM) binaries are provided for this release due to a broken toolchain. One of the providers of Python packages for Windows (TM) is your best bet.

15.64.1 Highlights

Details of these improvements can be found below.

- The datetime64 type is now timezone naive.
- A dtype parameter has been added to `randint`.
- Improved detection of two arrays possibly sharing memory.
- Automatic bin size estimation for `np.histogram`.
- Speed optimization of A @ A.T and dot(A, A.T).
- New function `np.moveaxis` for reordering array axes.

15.64.2 Build System Changes

- Numpy now uses `setuptools` for its builds instead of plain distutils. This fixes usage of `install_requires='numpy'` in the `setup.py` files of projects that depend on Numpy (see gh-6551). It potentially affects the way that build/install methods for Numpy itself behave though. Please report any unexpected behavior on the Numpy issue tracker.
- Bento build support and related files have been removed.
- Single file build support and related files have been removed.

15.64.3 Future Changes

The following changes are scheduled for Numpy 1.12.0.

- Support for Python 2.6, 3.2, and 3.3 will be dropped.
- Relaxed stride checking will become the default. See the 1.8.0 release notes for a more extended discussion of what this change implies.
- The behavior of the datetime64 "not a time" (NaT) value will be changed to match that of floating point "not a number" (NaN) values: all comparisons involving NaT will return False, except for NaT != NaT which will return True.
- Indexing with floats will raise IndexError, e.g., a[0, 0.0].
- Indexing with non-integer array_like will raise `IndexError`, e.g., `a['1', '2']`
- Indexing with multiple ellipsis will raise `IndexError`, e.g., `a[..., ...]`.
- Non-integers used as index values will raise `TypeError`, e.g., in `reshape`, `take`, and specifying reduce axis.

In a future release the following changes will be made.

- The `rand` function exposed in `numpy.testing` will be removed. That function is left over from early Numpy and was implemented using the Python random module. The random number generators from `numpy.random` should be used instead.
- The `ndarray.view` method will only allow c_contiguous arrays to be viewed using a dtype of different size causing the last dimension to change. That differs from the current behavior where arrays that are f_contiguous but not c_contiguous can be viewed as a dtype type of different size causing the first dimension to change.
- Slicing a `MaskedArray` will return views of both data **and** mask. Currently the mask is copy-on-write and changes to the mask in the slice do not propagate to the original mask. See the FutureWarnings section below for details.

15.64.4 Compatibility notes

datetime64 changes

In prior versions of NumPy the experimental datetime64 type always stored times in UTC. By default, creating a datetime64 object from a string or printing it would convert from or to local time:

```
# old behavior
>>> np.datetime64('2000-01-01T00:00:00')
numpy.datetime64('2000-01-01T00:00:00-0800')  # note the timezone offset -08:00
```

A consensus of datetime64 users agreed that this behavior is undesirable and at odds with how datetime64 is usually used (e.g., by pandas). For most use cases, a timezone naive datetime type is preferred, similar to the `datetime.datetime` type in the Python standard library. Accordingly, datetime64 no longer assumes that input is in local time, nor does it print local times:

```
>>> np.datetime64('2000-01-01T00:00:00')
numpy.datetime64('2000-01-01T00:00:00')
```

For backwards compatibility, datetime64 still parses timezone offsets, which it handles by converting to UTC. However, the resulting datetime is timezone naive:

```
>>> np.datetime64('2000-01-01T00:00:00-08')
DeprecationWarning: parsing timezone aware datetimes is deprecated;
this will raise an error in the future
numpy.datetime64('2000-01-01T08:00:00')
```

As a corollary to this change, we no longer prohibit casting between datetimes with date units and datetimes with time units. With timezone naive datetimes, the rule for casting from dates to times is no longer ambiguous.

`linalg.norm` return type changes

The return type of the `linalg.norm` function is now floating point without exception. Some of the norm types previously returned integers.

polynomial fit changes

The various fit functions in the numpy polynomial package no longer accept non-integers for degree specification.

np.dot now raises `TypeError` instead of `ValueError`

This behaviour mimics that of other functions such as `np.inner`. If the two arguments cannot be cast to a common type, it could have raised a `TypeError` or `ValueError` depending on their order. Now, `np.dot` will now always raise a `TypeError`.

FutureWarning to changed behavior

- In `np.lib.split` an empty array in the result always had dimension `(0,)` no matter the dimensions of the array being split. This has been changed so that the dimensions will be preserved. A `FutureWarning` for this change has been in place since Numpy 1.9 but, due to a bug, sometimes no warning was raised and the dimensions were already preserved.

`%` and `//` operators

These operators are implemented with the `remainder` and `floor_divide` functions respectively. Those functions are now based around `fmod` and are computed together so as to be compatible with each other and with the Python versions for float types. The results should be marginally more accurate or outright bug fixes compared to the previous results, but they may differ significantly in cases where roundoff makes a difference in the integer returned by `floor_divide`. Some corner cases also change, for instance, NaN is always returned for both functions when the divisor is zero, `divmod(1.0, inf)` returns `(0.0, 1.0)` except on MSVC 2008, and `divmod(-1.0, inf)` returns `(-1.0, inf)`.

C API

Removed the `check_return` and `inner_loop_selector` members of the `PyUFuncObject` struct (replacing them with `reserved` slots to preserve struct layout). These were never used for anything, so it's unlikely that any third-party code is using them either, but we mention it here for completeness.

object dtype detection for old-style classes

In python 2, objects which are instances of old-style user-defined classes no longer automatically count as 'object' type in the dtype-detection handler. Instead, as in python 3, they may potentially count as sequences, but only if they define both a *__len__* and a *__getitem__* method. This fixes a segfault and inconsistency between python 2 and 3.

15.64.5 New Features

- `np.histogram` now provides plugin estimators for automatically estimating the optimal number of bins. Passing one of ['auto', 'fd', 'scott', 'rice', 'sturges'] as the argument to 'bins' results in the corresponding estimator being used.
- A benchmark suite using Airspeed Velocity has been added, converting the previous vbench-based one. You can run the suite locally via `python runtests.py --bench`. For more details, see `benchmarks/README.rst`.
- A new function `np.shares_memory` that can check exactly whether two arrays have memory overlap is added. `np.may_share_memory` also now has an option to spend more effort to reduce false positives.
- `SkipTest` and `KnownFailureException` exception classes are exposed in the `numpy.testing` namespace. Raise them in a test function to mark the test to be skipped or mark it as a known failure, respectively.
- `f2py.compile` has a new `extension` keyword parameter that allows the fortran extension to be specified for generated temp files. For instance, the files can be specifies to be `*.f90`. The `verbose` argument is also activated, it was previously ignored.
- A `dtype` parameter has been added to `np.random.randint` Random ndarrays of the following types can now be generated:
  - `np.bool_`,
  - `np.int8`, `np.uint8`,

  - `np.int16`, `np.uint16`,
  - `np.int32`, `np.uint32`,
  - `np.int64`, `np.uint64`,
  - `np.int_ ``, ``np.intp`

  The specification is by precision rather than by C type. Hence, on some platforms `np.int64` may be a `long` instead of `long long` even if the specified dtype is `long long` because the two may have the same precision. The resulting type depends on which C type numpy uses for the given precision. The byteorder specification is also ignored, the generated arrays are always in native byte order.

- A new `np.moveaxis` function allows for moving one or more array axes to a new position by explicitly providing source and destination axes. This function should be easier to use than the current `rollaxis` function as well as providing more functionality.
- The `deg` parameter of the various `numpy.polynomial` fits has been extended to accept a list of the degrees of the terms to be included in the fit, the coefficients of all other terms being constrained to zero. The change is backward compatible, passing a scalar `deg` will behave as before.
- A divmod function for float types modeled after the Python version has been added to the npy_math library.

15.64.6 Improvements

`np.gradient` now supports an `axis` argument

The `axis` parameter was added to `np.gradient` for consistency. It allows to specify over which axes the gradient is calculated.

`np.lexsort` now supports arrays with object data-type

The function now internally calls the generic `npy_amergesort` when the type does not implement a merge-sort kind of `argsort` method.

`np.ma.core.MaskedArray` now supports an `order` argument

When constructing a new `MaskedArray` instance, it can be configured with an `order` argument analogous to the one when calling `np.ndarray`. The addition of this argument allows for the proper processing of an `order` argument in several MaskedArray-related utility functions such as `np.ma.core.array` and `np.ma.core.asarray`.

Memory and speed improvements for masked arrays

Creating a masked array with `mask=True` (resp. `mask=False`) now uses `np.ones` (resp. `np.zeros`) to create the mask, which is faster and avoid a big memory peak. Another optimization was done to avoid a memory peak and useless computations when printing a masked array.

`ndarray.tofile` now uses fallocate on linux

The function now uses the fallocate system call to reserve sufficient disk space on file systems that support it.

Optimizations for operations of the form `A.T @ A` and `A @ A.T`

Previously, `gemm` BLAS operations were used for all matrix products. Now, if the matrix product is between a matrix and its transpose, it will use `syrk` BLAS operations for a performance boost. This optimization has been extended to `@`, `numpy.dot`, `numpy.inner`, and `numpy.matmul`.

Note: Requires the transposed and non-transposed matrices to share data.

`np.testing.assert_warns` can now be used as a context manager

This matches the behavior of `assert_raises`.

Speed improvement for np.random.shuffle

`np.random.shuffle` is now much faster for 1d ndarrays.

15.64.7 Changes

Pyrex support was removed from `numpy.distutils`

The method `build_src.generate_a_pyrex_source` will remain available; it has been monkeypatched by users to support Cython instead of Pyrex. It's recommended to switch to a better supported method of build Cython extensions though.

`np.broadcast` can now be called with a single argument

The resulting object in that case will simply mimic iteration over a single array. This change obsoletes distinctions like

if len(x) == 1:

    shape = x[0].shape

else:

    shape = np.broadcast(*x).shape

Instead, `np.broadcast` can be used in all cases.

`np.trace` now respects array subclasses

This behaviour mimics that of other functions such as `np.diagonal` and ensures, e.g., that for masked arrays `np.trace(ma)` and `ma.trace()` give the same result.

`np.dot` now raises `TypeError` instead of `ValueError`

This behaviour mimics that of other functions such as `np.inner`. If the two arguments cannot be cast to a common type, it could have raised a `TypeError` or `ValueError` depending on their order. Now, `np.dot` will now always raise a `TypeError`.

`linalg.norm` return type changes

The `linalg.norm` function now does all its computations in floating point and returns floating results. This change fixes bugs due to integer overflow and the failure of abs with signed integers of minimum value, e.g., int8(-128). For consistency, floats are used even where an integer might work.

15.64.8 Deprecations

Views of arrays in Fortran order

The F_CONTIGUOUS flag was used to signal that views using a dtype that changed the element size would change the first index. This was always problematical for arrays that were both F_CONTIGUOUS and C_CONTIGUOUS because C_CONTIGUOUS took precedence. Relaxed stride checking results in more such dual contiguous arrays and breaks some existing code as a result. Note that this also affects changing the dtype by assigning to the dtype attribute of an array. The aim of this deprecation is to restrict views to C_CONTIGUOUS arrays at some future time. A work around that is backward compatible is to use `a.T.view(...).T` instead. A parameter may also be added to the view method to explicitly ask for Fortran order views, but that will not be backward compatible.

Invalid arguments for array ordering

It is currently possible to pass in arguments for the `order` parameter in methods like `array.flatten` or `array.ravel` that were not one of the following: 'C', 'F', 'A', 'K' (note that all of these possible values are both unicode and case insensitive). Such behavior will not be allowed in future releases.

Random number generator in the `testing` namespace

The Python standard library random number generator was previously exposed in the `testing` namespace as `testing.rand`. Using this generator is not recommended and it will be removed in a future release. Use generators from `numpy.random` namespace instead.

Random integer generation on a closed interval

In accordance with the Python C API, which gives preference to the half-open interval over the closed one, `np.random.random_integers` is being deprecated in favor of calling `np.random.randint`, which has been enhanced with the `dtype` parameter as described under "New Features". However, `np.random.random_integers` will not be removed anytime soon.

15.64.9 FutureWarnings

Assigning to slices/views of `MaskedArray`

Currently a slice of a masked array contains a view of the original data and a copy-on-write view of the mask. Consequently, any changes to the slice's mask will result in a copy of the original mask being made and that new mask being changed rather than the original. For example, if we make a slice of the original like so, `view = original[:]`, then modifications to the data in one array will affect the data of the other but, because the mask will be copied during assignment operations, changes to the mask will remain local. A similar situation occurs when explicitly constructing a masked array using `MaskedArray(data, mask)`, the returned array will contain a view of `data` but the mask will be a copy-on-write view of `mask`.

In the future, these cases will be normalized so that the data and mask arrays are treated the same way and modifications to either will propagate between views. In 1.11, numpy will issue a `MaskedArrayFutureWarning` warning whenever user code modifies the mask of a view that in the future may cause values to propagate back to the original. To silence these warnings and make your code robust against the upcoming changes, you have two options: if you want to keep the current behavior, call `masked_view.unshare_mask()` before modifying the mask. If you want to get the future behavior early, use `masked_view._sharedmask = False`. However, note that setting the `_sharedmask` attribute will break following explicit calls to `masked_view.unshare_mask()`.

15.65 NumPy 1.10.4 Release Notes

This release is a bugfix source release motivated by a segfault regression. No windows binaries are provided for this release, as there appear to be bugs in the toolchain we use to generate those files. Hopefully that problem will be fixed for the next release. In the meantime, we suggest using one of the providers of windows binaries.

15.65.1 Compatibility notes

- The trace function now calls the trace method on subclasses of ndarray, except for matrix, for which the current behavior is preserved. This is to help with the units package of AstroPy and hopefully will not cause problems.

15.65.2 Issues Fixed

- gh-6922 BUG: numpy.recarray.sort segfaults on Windows.
- gh-6937 BUG: busday_offset does the wrong thing with modifiedpreceding roll.
- gh-6949 BUG: Type is lost when slicing a subclass of recarray.

15.65.3 Merged PRs

The following PRs have been merged into 1.10.4. When the PR is a backport, the PR number for the original PR against master is listed.

- gh-6840 TST: Update travis testing script in 1.10.x
- gh-6843 BUG: Fix use of python 3 only FileNotFoundError in test_f2py.
- gh-6884 REL: Update pavement.py and setup.py to reflect current version.
- gh-6916 BUG: Fix test_f2py so it runs correctly in runtests.py.
- gh-6924 BUG: Fix segfault gh-6922.

- gh-6942 Fix datetime roll='modifiedpreceding' bug.
- gh-6943 DOC,BUG: Fix some latex generation problems.
- gh-6950 BUG trace is not subclass aware, np.trace(ma) != ma.trace().
- gh-6952 BUG recarray slices should preserve subclass.

15.66 NumPy 1.10.3 Release Notes

N/A this release did not happen due to various screwups involving PyPI.

15.67 NumPy 1.10.2 Release Notes

This release deals with a number of bugs that turned up in 1.10.1 and adds various build and release improvements.

Numpy 1.10.1 supports Python 2.6 - 2.7 and 3.2 - 3.5.

15.67.1 Compatibility notes

Relaxed stride checking is no longer the default

There were back compatibility problems involving views changing the dtype of multidimensional Fortran arrays that need to be dealt with over a longer timeframe.

Fix swig bug in `numpy.i`

Relaxed stride checking revealed a bug in `array_is_fortran(a)`, that was using PyArray_ISFORTRAN to check for Fortran contiguity instead of PyArray_IS_F_CONTIGUOUS. You may want to regenerate swigged files using the updated numpy.i

Deprecate views changing dimensions in fortran order

This deprecates assignment of a new descriptor to the dtype attribute of a non-C-contiguous array if it result in changing the shape. This effectively bars viewing a multidimensional Fortran array using a dtype that changes the element size along the first axis.

The reason for the deprecation is that, when relaxed strides checking is enabled, arrays that are both C and Fortran contiguous are always treated as C contiguous which breaks some code that depended the two being mutually exclusive for non-scalar arrays of ndim > 1. This deprecation prepares the way to always enable relaxed stride checking.

15.67.2 Issues Fixed

- gh-6019 Masked array repr fails for structured array with multi-dimensional column.
- gh-6462 Median of empty array produces IndexError.
- gh-6467 Performance regression for record array access.
- gh-6468 numpy.interp uses 'left' value even when x[0]==xp[0].
- gh-6475 np.allclose returns a memmap when one of its arguments is a memmap.
- gh-6491 Error in broadcasting stride_tricks array.
- gh-6495 Unrecognized command line option '-ffpe-summary' in gfortran.
- gh-6497 Failure of reduce operation on recarrays.
- gh-6498 Mention change in default casting rule in 1.10 release notes.
- gh-6530 The partition function errors out on empty input.
- gh-6532 numpy.inner return wrong inaccurate value sometimes.
- gh-6563 Intent(out) broken in recent versions of f2py.
- gh-6569 Cannot run tests after 'python setup.py build_ext -i'
- gh-6572 Error in broadcasting stride_tricks array component.
- gh-6575 BUG: Split produces empty arrays with wrong number of dimensions
- gh-6590 Fortran Array problem in numpy 1.10.
- gh-6602 Random __all__ missing choice and dirichlet.
- gh-6611 ma.dot no longer always returns a masked array in 1.10.
- gh-6618 NPY_FORTRANORDER in make_fortran() in numpy.i
- gh-6636 Memory leak in nested dtypes in numpy.recarray
- gh-6641 Subsetting recarray by fields yields a structured array.
- gh-6667 ma.make_mask handles ma.nomask input incorrectly.
- gh-6675 Optimized blas detection broken in master and 1.10.
- gh-6678 Getting unexpected error from: X.dtype = complex (or Y = X.view(complex))
- gh-6718 f2py test fail in pip installed numpy-1.10.1 in virtualenv.
- gh-6719 Error compiling Cython file: Pythonic division not allowed without gil.
- gh-6771 Numpy.rec.fromarrays losing dtype metadata between versions 1.9.2 and 1.10.1
- gh-6781 The travis-ci script in maintenance/1.10.x needs fixing.
- gh-6807 Windows testing errors for 1.10.2

15.67.3 Merged PRs

The following PRs have been merged into 1.10.2. When the PR is a backport, the PR number for the original PR against master is listed.

- gh-5773 MAINT: Hide testing helper tracebacks when using them with pytest.
- gh-6094 BUG: Fixed a bug with string representation of masked structured arrays.
- gh-6208 MAINT: Speedup field access by removing unneeded safety checks.
- gh-6460 BUG: Replacing the os.environ.clear by less invasive procedure.
- gh-6470 BUG: Fix AttributeError in numpy distutils.
- gh-6472 MAINT: Use Python 3.5 instead of 3.5-dev for travis 3.5 testing.
- gh-6474 REL: Update Paver script for sdist and auto-switch test warnings.
- gh-6478 BUG: Fix Intel compiler flags for OS X build.
- gh-6481 MAINT: LIBPATH with spaces is now supported Python 2.7+ and Win32.
- gh-6487 BUG: Allow nested use of parameters in definition of arrays in f2py.
- gh-6488 BUG: Extend common blocks rather than overwriting in f2py.
- gh-6499 DOC: Mention that default casting for inplace operations has changed.
- gh-6500 BUG: Recarrays viewed as subarrays don't convert to np.record type.
- gh-6501 REL: Add "make upload" command for built docs, update "make dist".
- gh-6526 BUG: Fix use of __doc__ in setup.py for -OO mode.
- gh-6527 BUG: Fix the IndexError when taking the median of an empty array.
- gh-6537 BUG: Make ma.atleast_* with scalar argument return arrays.
- gh-6538 BUG: Fix ma.masked_values does not shrink mask if requested.
- gh-6546 BUG: Fix inner product regression for non-contiguous arrays.
- gh-6553 BUG: Fix partition and argpartition error for empty input.
- gh-6556 BUG: Error in broadcast_arrays with as_strided array.
- gh-6558 MAINT: Minor update to "make upload" doc build command.
- gh-6562 BUG: Disable view safety checks in recarray.
- gh-6567 BUG: Revert some import * fixes in f2py.
- gh-6574 DOC: Release notes for Numpy 1.10.2.
- gh-6577 BUG: Fix for #6569, allowing build_ext –inplace
- gh-6579 MAINT: Fix mistake in doc upload rule.
- gh-6596 BUG: Fix swig for relaxed stride checking.
- gh-6606 DOC: Update 1.10.2 release notes.
- gh-6614 BUG: Add choice and dirichlet to numpy.random.__all__.
- gh-6621 BUG: Fix swig make_fortran function.
- gh-6628 BUG: Make allclose return python bool.
- gh-6642 BUG: Fix memleak in _convert_from_dict.

- gh-6643 ENH: make recarray.getitem return a recarray.
- gh-6653 BUG: Fix ma dot to always return masked array.
- gh-6668 BUG: ma.make_mask should always return nomask for nomask argument.
- gh-6686 BUG: Fix a bug in assert_string_equal.
- gh-6695 BUG: Fix removing tempdirs created during build.
- gh-6697 MAINT: Fix spurious semicolon in macro definition of PyArray_FROM_OT.
- gh-6698 TST: test np.rint bug for large integers.
- gh-6717 BUG: Readd fallback CBLAS detection on linux.
- gh-6721 BUG: Fix for #6719.
- gh-6726 BUG: Fix bugs exposed by relaxed stride rollback.
- gh-6757 BUG: link cblas library if cblas is detected.
- gh-6756 TST: only test f2py, not f2py2.7 etc, fixes #6718.
- gh-6747 DEP: Deprecate changing shape of non-C-contiguous array via descr.
- gh-6775 MAINT: Include from __future__ boilerplate in some files missing it.
- gh-6780 BUG: metadata is not copied to base_dtype.
- gh-6783 BUG: Fix travis ci testing for new google infrastructure.
- gh-6785 BUG: Quick and dirty fix for interp.
- gh-6813 TST,BUG: Make test_mvoid_multidim_print work for 32 bit systems.
- gh-6817 BUG: Disable 32-bit msvc9 compiler optimizations for npy_rint.
- gh-6819 TST: Fix test_mvoid_multidim_print failures on Python 2.x for Windows.

Initial support for mingwpy was reverted as it was causing problems for non-windows builds.

- gh-6536 BUG: Revert gh-5614 to fix non-windows build problems

A fix for np.lib.split was reverted because it resulted in "fixing" behavior that will be present in the Numpy 1.11 and that was already present in Numpy 1.9. See the discussion of the issue at gh-6575 for clarification.

- gh-6576 BUG: Revert gh-6376 to fix split behavior for empty arrays.

Relaxed stride checking was reverted. There were back compatibility problems involving views changing the dtype of multidimensional Fortran arrays that need to be dealt with over a longer timeframe.

- gh-6735 MAINT: Make no relaxed stride checking the default for 1.10.

15.67.4 Notes

A bug in the Numpy 1.10.1 release resulted in exceptions being raised for `RuntimeWarning` and `DeprecationWarning` in projects depending on Numpy. That has been fixed.

15.68 NumPy 1.10.1 Release Notes

This release deals with a few build problems that showed up in 1.10.0. Most users would not have seen these problems. The differences are:

- Compiling with msvc9 or msvc10 for 32 bit Windows now requires SSE2. This was the easiest fix for what looked to be some miscompiled code when SSE2 was not used. If you need to compile for 32 bit Windows systems without SSE2 support, mingw32 should still work.
- Make compiling with VS2008 python2.7 SDK easier
- Change Intel compiler options so that code will also be generated to support systems without SSE4.2.
- Some _config test functions needed an explicit integer return in order to avoid the openSUSE rpmlinter erring out.
- We ran into a problem with pipy not allowing reuse of filenames and a resulting proliferation of ..*.postN releases. Not only were the names getting out of hand, some packages were unable to work with the postN suffix.

Numpy 1.10.1 supports Python 2.6 - 2.7 and 3.2 - 3.5.

Commits:

45a3d84 DEP: Remove warning for *full* when dtype is set. 0c1a5df BLD: import setuptools to allow compile with VS2008 python2.7 sdk 04211c6 BUG: mask nan to 1 in ordered compare 826716f DOC: Document the reason msvc requires SSE2 on 32 bit platforms. 49fa187 BLD: enable SSE2 for 32-bit msvc 9 and 10 compilers dcbc4cc MAINT: remove Wreturn-type warnings from config checks d6564cb BLD: do not build exclusively for SSE4.2 processors 15cb66f BLD: do not build exclusively for SSE4.2 processors c38bc08 DOC: fix var. reference in percentile docstring 78497f4 DOC: Sync 1.10.0-notes.rst in 1.10.x branch with master.

15.69 NumPy 1.10.0 Release Notes

This release supports Python 2.6 - 2.7 and 3.2 - 3.5.

15.69.1 Highlights

- numpy.distutils now supports parallel compilation via the –parallel/-j argument passed to setup.py build
- numpy.distutils now supports additional customization via site.cfg to control compilation parameters, i.e. runtime libraries, extra linking/compilation flags.
- Addition of *np.linalg.multi_dot*: compute the dot product of two or more arrays in a single function call, while automatically selecting the fastest evaluation order.
- The new function *np.stack* provides a general interface for joining a sequence of arrays along a new axis, complementing *np.concatenate* for joining along an existing axis.
- Addition of *nanprod* to the set of nanfunctions.
- Support for the '@' operator in Python 3.5.

15.69.2 Dropped Support

- The _dotblas module has been removed. CBLAS Support is now in Multiarray.
- The testcalcs.py file has been removed.
- The polytemplate.py file has been removed.
- npy_PyFile_Dup and npy_PyFile_DupClose have been removed from npy_3kcompat.h.
- splitcmdline has been removed from numpy/distutils/exec_command.py.
- try_run and get_output have been removed from numpy/distutils/command/config.py
- The a._format attribute is no longer supported for array printing.
- Keywords `skiprows` and `missing` removed from np.genfromtxt.
- Keyword `old_behavior` removed from np.correlate.

15.69.3 Future Changes

- In array comparisons like `arr1 == arr2`, many corner cases involving strings or structured dtypes that used to return scalars now issue `FutureWarning` or `DeprecationWarning`, and in the future will be change to either perform elementwise comparisons or raise an error.
- In `np.lib.split` an empty array in the result always had dimension `(0,)` no matter the dimensions of the array being split. In Numpy 1.11 that behavior will be changed so that the dimensions will be preserved. A `FutureWarning` for this change has been in place since Numpy 1.9 but, due to a bug, sometimes no warning was raised and the dimensions were already preserved.
- The SafeEval class will be removed in Numpy 1.11.
- The alterdot and restoredot functions will be removed in Numpy 1.11.

See below for more details on these changes.

15.69.4 Compatibility notes

Default casting rule change

Default casting for inplace operations has changed to `'same_kind'`. For instance, if n is an array of integers, and f is an array of floats, then `n += f` will result in a `TypeError`, whereas in previous Numpy versions the floats would be silently cast to ints. In the unlikely case that the example code is not an actual bug, it can be updated in a backward compatible way by rewriting it as `np.add(n, f, out=n, casting='unsafe')`. The old `'unsafe'` default has been deprecated since Numpy 1.7.

numpy version string

The numpy version string for development builds has been changed from `x.y.z.dev-githash` to `x.y.z.dev0+githash` (note the +) in order to comply with PEP 440.

relaxed stride checking

NPY_RELAXED_STRIDE_CHECKING is now true by default.

UPDATE: In 1.10.2 the default value of NPY_RELAXED_STRIDE_CHECKING was changed to false for back compatibility reasons. More time is needed before it can be made the default. As part of the roadmap a deprecation of dimension changing views of f_contiguous not c_contiguous arrays was also added.

Concatenation of 1d arrays along any but `axis=0` raises `IndexError`

Using axis != 0 has raised a DeprecationWarning since NumPy 1.7, it now raises an error.

np.ravel, *np.diagonal* and *np.diag* now preserve subtypes

There was inconsistent behavior between *x.ravel()* and *np.ravel(x)*, as well as between *x.diagonal()* and *np.diagonal(x)*, with the methods preserving subtypes while the functions did not. This has been fixed and the functions now behave like the methods, preserving subtypes except in the case of matrices. Matrices are special cased for backward compatibility and still return 1-D arrays as before. If you need to preserve the matrix subtype, use the methods instead of the functions.

rollaxis and *swapaxes* always return a view

Previously, a view was returned except when no change was made in the order of the axes, in which case the input array was returned. A view is now returned in all cases.

nonzero now returns base ndarrays

Previously, an inconsistency existed between 1-D inputs (returning a base ndarray) and higher dimensional ones (which preserved subclasses). Behavior has been unified, and the return will now be a base ndarray. Subclasses can still override this behavior by providing their own *nonzero* method.

C API

The changes to *swapaxes* also apply to the *PyArray_SwapAxes* C function, which now returns a view in all cases.

The changes to *nonzero* also apply to the *PyArray_Nonzero* C function, which now returns a base ndarray in all cases.

The dtype structure (PyArray_Descr) has a new member at the end to cache its hash value. This shouldn't affect any well-written applications.

The change to the concatenation function DeprecationWarning also affects PyArray_ConcatenateArrays,

recarray field return types

Previously the returned types for recarray fields accessed by attribute and by index were inconsistent, and fields of string type were returned as chararrays. Now, fields accessed by either attribute or indexing will return an ndarray for fields of non-structured type, and a recarray for fields of structured type. Notably, this affect recarrays containing strings with whitespace, as trailing whitespace is trimmed from chararrays but kept in ndarrays of string type. Also, the dtype.type of nested structured fields is now inherited.

recarray views

Viewing an ndarray as a recarray now automatically converts the dtype to np.record. See new record array documentation. Additionally, viewing a recarray with a non-structured dtype no longer converts the result's type to ndarray - the result will remain a recarray.

'out' keyword argument of ufuncs now accepts tuples of arrays

When using the 'out' keyword argument of a ufunc, a tuple of arrays, one per ufunc output, can be provided. For ufuncs with a single output a single array is also a valid 'out' keyword argument. Previously a single array could be provided in the 'out' keyword argument, and it would be used as the first output for ufuncs with multiple outputs, is deprecated, and will result in a *DeprecationWarning* now and an error in the future.

byte-array indices now raises an IndexError

Indexing an ndarray using a byte-string in Python 3 now raises an IndexError instead of a ValueError.

Masked arrays containing objects with arrays

For such (rare) masked arrays, getting a single masked item no longer returns a corrupted masked array, but a fully masked version of the item.

Median warns and returns nan when invalid values are encountered

Similar to mean, median and percentile now emits a Runtime warning and returns *NaN* in slices where a *NaN* is present. To compute the median or percentile while ignoring invalid values use the new *nanmedian* or *nanpercentile* functions.

Functions available from numpy.ma.testutils have changed

All functions from numpy.testing were once available from numpy.ma.testutils but not all of them were redefined to work with masked arrays. Most of those functions have now been removed from numpy.ma.testutils with a small subset retained in order to preserve backward compatibility. In the long run this should help avoid mistaken use of the wrong functions, but it may cause import problems for some.

15.69.5 New Features

Reading extra flags from site.cfg

Previously customization of compilation of dependency libraries and numpy itself was only accomblishable via code changes in the distutils package. Now numpy.distutils reads in the following extra flags from each group of the *site.cfg*:

- **`runtime_library_dirs/rpath`, sets runtime library directories to override**

  `LD_LIBRARY_PATH`
- `extra_compile_args`, add extra flags to the compilation of sources
- `extra_link_args`, add extra flags when linking libraries

This should, at least partially, complete user customization.

np.cbrt to compute cube root for real floats

np.cbrt wraps the C99 cube root function *cbrt*. Compared to *np.power(x, 1./3.)* it is well defined for negative real floats and a bit faster.

By passing *–parallel=n* or *-j n* to *setup.py build* the compilation of extensions is now performed in *n* parallel processes. The parallelization is limited to files within one extension so projects using Cython will not profit because it builds extensions from single files.

genfromtxt has a new `max_rows` argument

A `max_rows` argument has been added to *genfromtxt* to limit the number of rows read in a single call. Using this functionality, it is possible to read in multiple arrays stored in a single file by making repeated calls to the function.

New function *np.broadcast_to* for invoking array broadcasting

np.broadcast_to manually broadcasts an array to a given shape according to numpy's broadcasting rules. The functionality is similar to broadcast_arrays, which in fact has been rewritten to use broadcast_to internally, but only a single array is necessary.

New context manager *clear_and_catch_warnings* for testing warnings

When Python emits a warning, it records that this warning has been emitted in the module that caused the warning, in a module attribute `__warningregistry__`. Once this has happened, it is not possible to emit the warning again, unless you clear the relevant entry in `__warningregistry__`. This makes is hard and fragile to test warnings, because if your test comes after another that has already caused the warning, you will not be able to emit the warning or test it. The context manager `clear_and_catch_warnings` clears warnings from the module registry on entry and resets them on exit, meaning that warnings can be re-raised.

cov has new `fweights` and `aweights` arguments

The `fweights` and `aweights` arguments add new functionality to covariance calculations by applying two types of weighting to observation vectors. An array of `fweights` indicates the number of repeats of each observation vector, and an array of `aweights` provides their relative importance or probability.

Support for the '@' operator in Python 3.5+

Python 3.5 adds support for a matrix multiplication operator '@' proposed in PEP465. Preliminary support for that has been implemented, and an equivalent function `matmul` has also been added for testing purposes and use in earlier Python versions. The function is preliminary and the order and number of its optional arguments can be expected to change.

New argument `norm` to fft functions

The default normalization has the direct transforms unscaled and the inverse transforms are scaled by $1/n$. It is possible to obtain unitary transforms by setting the keyword argument `norm` to `"ortho"` (default is *None*) so that both direct and inverse transforms will be scaled by $1/sqrtn$.

15.69.6 Improvements

np.digitize using binary search

np.digitize is now implemented in terms of *np.searchsorted*. This means that a binary search is used to bin the values, which scales much better for larger number of bins than the previous linear search. It also removes the requirement for the input array to be 1-dimensional.

np.poly now casts integer inputs to float

np.poly will now cast 1-dimensional input arrays of integer type to double precision floating point, to prevent integer overflow when computing the monic polynomial. It is still possible to obtain higher precision results by passing in an array of object type, filled e.g. with Python ints.

np.interp can now be used with periodic functions

np.interp now has a new parameter *period* that supplies the period of the input data *xp*. In such case, the input data is properly normalized to the given period and one end point is added to each extremity of *xp* in order to close the previous and the next period cycles, resulting in the correct interpolation behavior.

np.pad supports more input types for `pad_width` and `constant_values`

`constant_values` parameters now accepts NumPy arrays and float values. NumPy arrays are supported as input for `pad_width`, and an exception is raised if its values are not of integral type.

np.argmax and *np.argmin* now support an `out` argument

The `out` parameter was added to *np.argmax* and *np.argmin* for consistency with *ndarray.argmax* and *ndarray.argmin*. The new parameter behaves exactly as it does in those methods.

More system C99 complex functions detected and used

All of the functions `in complex.h` are now detected. There are new fallback implementations of the following functions.

- npy_ctan,
- npy_cacos, npy_casin, npy_catan
- npy_ccosh, npy_csinh, npy_ctanh,
- npy_cacosh, npy_casinh, npy_catanh

As a result of these improvements, there will be some small changes in returned values, especially for corner cases.

np.loadtxt support for the strings produced by the `float.hex` method

The strings produced by `float.hex` look like `0x1.921fb54442d18p+1`, so this is not the hex used to represent unsigned integer types.

np.isclose properly handles minimal values of integer dtypes

In order to properly handle minimal values of integer types, *np.isclose* will now cast to the float dtype during comparisons. This aligns its behavior with what was provided by *np.allclose*.

np.allclose uses *np.isclose* internally.

np.allclose now uses *np.isclose* internally and inherits the ability to compare NaNs as equal by setting `equal_nan=True`. Subclasses, such as *np.ma.MaskedArray*, are also preserved now.

np.genfromtxt now handles large integers correctly

np.genfromtxt now correctly handles integers larger than `2**31-1` on 32-bit systems and larger than `2**63-1` on 64-bit systems (it previously crashed with an `OverflowError` in these cases). Integers larger than `2**63-1` are converted to floating-point values.

np.load, *np.save* have pickle backward compatibility flags

The functions *np.load* and *np.save* have additional keyword arguments for controlling backward compatibility of pickled Python objects. This enables Numpy on Python 3 to load npy files containing object arrays that were generated on Python 2.

MaskedArray support for more complicated base classes

Built-in assumptions that the baseclass behaved like a plain array are being removed. In particular, setting and getting elements and ranges will respect baseclass overrides of `__setitem__` and `__getitem__`, and arithmetic will respect overrides of `__add__`, `__sub__`, etc.

15.69.7 Changes

dotblas functionality moved to multiarray

The cblas versions of dot, inner, and vdot have been integrated into the multiarray module. In particular, vdot is now a multiarray function, which it was not before.

stricter check of gufunc signature compliance

Inputs to generalized universal functions are now more strictly checked against the function's signature: all core dimensions are now required to be present in input arrays; core dimensions with the same label must have the exact same size; and output core dimension's must be specified, either by a same label input core dimension or by a passed-in output array.

views returned from *np.einsum* are writeable

Views returned by *np.einsum* will now be writeable whenever the input array is writeable.

np.argmin skips NaT values

np.argmin now skips NaT values in datetime64 and timedelta64 arrays, making it consistent with *np.min*, *np.argmax* and *np.max*.

15.69.8 Deprecations

Array comparisons involving strings or structured dtypes

Normally, comparison operations on arrays perform elementwise comparisons and return arrays of booleans. But in some corner cases, especially involving strings are structured dtypes, NumPy has historically returned a scalar instead. For example:

```
### Current behaviour

np.arange(2) == "foo"
# -> False

np.arange(2) < "foo"
# -> True on Python 2, error on Python 3

np.ones(2, dtype="i4,i4") == np.ones(2, dtype="i4,i4,i4")
# -> False
```

Continuing work started in 1.9, in 1.10 these comparisons will now raise `FutureWarning` or `DeprecationWarning`, and in the future they will be modified to behave more consistently with other comparison operations, e.g.:

```
### Future behaviour

np.arange(2) == "foo"
# -> array([False, False])

np.arange(2) < "foo"
# -> error, strings and numbers are not orderable

np.ones(2, dtype="i4,i4") == np.ones(2, dtype="i4,i4,i4")
# -> [False, False]
```

SafeEval

The SafeEval class in numpy/lib/utils.py is deprecated and will be removed in the next release.

alterdot, restoredot

The alterdot and restoredot functions no longer do anything, and are deprecated.

pkgload, PackageLoader

These ways of loading packages are now deprecated.

bias, ddof arguments to corrcoef

The values for the `bias` and `ddof` arguments to the `corrcoef` function canceled in the division implied by the correlation coefficient and so had no effect on the returned values.

We now deprecate these arguments to `corrcoef` and the masked array version `ma.corrcoef`.

Because we are deprecating the `bias` argument to `ma.corrcoef`, we also deprecate the use of the `allow_masked` argument as a positional argument, as its position will change with the removal of `bias`. `allow_masked` will in due course become a keyword-only argument.

dtype string representation changes

Since 1.6, creating a dtype object from its string representation, e.g. `'f4'`, would issue a deprecation warning if the size did not correspond to an existing type, and default to creating a dtype of the default size for the type. Starting with this release, this will now raise a `TypeError`.

The only exception is object dtypes, where both `'O4'` and `'O8'` will still issue a deprecation warning. This platform-dependent representation will raise an error in the next release.

In preparation for this upcoming change, the string representation of an object dtype, i.e. `np.dtype(object).str`, no longer includes the item size, i.e. will return `'|O'` instead of `'|O4'` or `'|O8'` as before.

15.70 NumPy 1.9.2 Release Notes

This is a bugfix only release in the 1.9.x series.

15.70.1 Issues fixed

- #5316: fix too large dtype alignment of strings and complex types
- #5424: fix ma.median when used on ndarrays
- #5481: Fix astype for structured array fields of different byte order
- #5354: fix segfault when clipping complex arrays
- #5524: allow np.argpartition on non ndarrays
- #5612: Fixes ndarray.fill to accept full range of uint64

- #5155: Fix loadtxt with comments=None and a string None data
- #4476: Masked array view fails if structured dtype has datetime component
- #5388: Make RandomState.set_state and RandomState.get_state threadsafe
- #5390: make seed, randint and shuffle threadsafe
- #5374: Fixed incorrect assert_array_almost_equal_nulp documentation
- #5393: Add support for ATLAS > 3.9.33.
- #5313: PyArray_AsCArray caused segfault for 3d arrays
- #5492: handle out of memory in rfftf
- #4181: fix a few bugs in the random.pareto docstring
- #5359: minor changes to linspace docstring
- #4723: fix a compile issues on AIX

15.71 NumPy 1.9.1 Release Notes

This is a bugfix only release in the 1.9.x series.

15.71.1 Issues fixed

- gh-5184: restore linear edge behaviour of gradient to as it was in < 1.9. The second order behaviour is available via the *edge_order* keyword
- gh-4007: workaround Accelerate sgemv crash on OSX 10.9
- gh-5100: restore object dtype inference from iterable objects without *len()*
- gh-5163: avoid gcc-4.1.2 (red hat 5) miscompilation causing a crash
- gh-5138: fix nanmedian on arrays containing inf
- gh-5240: fix not returning out array from ufuncs with subok=False set
- gh-5203: copy inherited masks in MaskedArray.__array_finalize__
- gh-2317: genfromtxt did not handle filling_values=0 correctly
- gh-5067: restore api of npy_PyFile_DupClose in python2
- gh-5063: cannot convert invalid sequence index to tuple
- gh-5082: Segmentation fault with argmin() on unicode arrays
- gh-5095: don't propagate subtypes from np.where
- gh-5104: np.inner segfaults with SciPy's sparse matrices
- gh-5251: Issue with fromarrays not using correct format for unicode arrays
- gh-5136: Import dummy_threading if importing threading fails
- gh-5148: Make numpy import when run with Python flag '-OO'
- gh-5147: Einsum double contraction in particular order causes ValueError
- gh-479: Make f2py work with intent(in out)

- gh-5170: Make python2 .npy files readable in python3
- gh-5027: Use 'll' as the default length specifier for long long
- gh-4896: fix build error with MSVC 2013 caused by C99 complex support
- gh-4465: Make PyArray_PutTo respect writeable flag
- gh-5225: fix crash when using arange on datetime without dtype set
- gh-5231: fix build in c99 mode

15.72 NumPy 1.9.0 Release Notes

This release supports Python 2.6 - 2.7 and 3.2 - 3.4.

15.72.1 Highlights

- Numerous performance improvements in various areas, most notably indexing and operations on small arrays are significantly faster. Indexing operations now also release the GIL.
- Addition of *nanmedian* and *nanpercentile* rounds out the nanfunction set.

15.72.2 Dropped Support

- The oldnumeric and numarray modules have been removed.
- The doc/pyrex and doc/cython directories have been removed.
- The doc/numpybook directory has been removed.
- The numpy/testing/numpytest.py file has been removed together with the importall function it contained.

15.72.3 Future Changes

- The numpy/polynomial/polytemplate.py file will be removed in NumPy 1.10.0.
- Default casting for inplace operations will change to 'same_kind' in Numpy 1.10.0. This will certainly break some code that is currently ignoring the warning.
- Relaxed stride checking will be the default in 1.10.0
- String version checks will break because, e.g., '1.9' > '1.10' is True. A NumpyVersion class has been added that can be used for such comparisons.
- The diagonal and diag functions will return writeable views in 1.10.0
- The *S* and/or *a* dtypes may be changed to represent Python strings instead of bytes, in Python 3 these two types are very different.

15.72.4 Compatibility notes

The diagonal and diag functions return readonly views.

In NumPy 1.8, the diagonal and diag functions returned readonly copies, in NumPy 1.9 they return readonly views, and in 1.10 they will return writeable views.

Special scalar float values don't cause upcast to double anymore

In previous numpy versions operations involving floating point scalars containing special values `NaN`, `Inf` and `-Inf` caused the result type to be at least `float64`. As the special values can be represented in the smallest available floating point type, the upcast is not performed anymore.

For example the dtype of:

```
np.array([1.], dtype=np.float32) * float('nan')
```

now remains `float32` instead of being cast to `float64`. Operations involving non-special values have not been changed.

Percentile output changes

If given more than one percentile to compute numpy.percentile returns an array instead of a list. A single percentile still returns a scalar. The array is equivalent to converting the list returned in older versions to an array via `np.array`.

If the `overwrite_input` option is used the input is only partially instead of fully sorted.

ndarray.tofile exception type

All `tofile` exceptions are now `IOError`, some were previously `ValueError`.

Invalid fill value exceptions

Two changes to numpy.ma.core._check_fill_value:

- When the fill value is a string and the array type is not one of 'OSUV', TypeError is raised instead of the default fill value being used.
- When the fill value overflows the array type, TypeError is raised instead of OverflowError.

Polynomial Classes no longer derived from PolyBase

This may cause problems with folks who depended on the polynomial classes being derived from PolyBase. They are now all derived from the abstract base class ABCPolyBase. Strictly speaking, there should be a deprecation involved, but no external code making use of the old baseclass could be found.

Using numpy.random.binomial may change the RNG state vs. numpy < 1.9

A bug in one of the algorithms to generate a binomial random variate has been fixed. This change will likely alter the number of random draws performed, and hence the sequence location will be different after a call to distribution.c::rk_binomial_btpe. Any tests which rely on the RNG being in a known state should be checked and/or updated as a result.

Random seed enforced to be a 32 bit unsigned integer

`np.random.seed` and `np.random.RandomState` now throw a `ValueError` if the seed cannot safely be converted to 32 bit unsigned integers. Applications that now fail can be fixed by masking the higher 32 bit values to zero: `seed = seed & 0xFFFFFFFF`. This is what is done silently in older versions so the random stream remains the same.

Argmin and argmax out argument

The `out` argument to `np.argmin` and `np.argmax` and their equivalent C-API functions is now checked to match the desired output shape exactly. If the check fails a `ValueError` instead of `TypeError` is raised.

Einsum

Remove unnecessary broadcasting notation restrictions. `np.einsum('ijk,j->ijk', A, B)` can also be written as `np.einsum('ij...,j->ij...', A, B)` (ellipsis is no longer required on 'j')

Indexing

The NumPy indexing has seen a complete rewrite in this version. This makes most advanced integer indexing operations much faster and should have no other implications. However some subtle changes and deprecations were introduced in advanced indexing operations:

- Boolean indexing into scalar arrays will always return a new 1-d array. This means that `array(1)[array(True)]` gives `array([1])` and not the original array.
- Advanced indexing into one dimensional arrays used to have (undocumented) special handling regarding repeating the value array in assignments when the shape of the value array was too small or did not match. Code using this will raise an error. For compatibility you can use `arr.flat[index] = values`, which uses the old code branch. (for example `a = np.ones(10); a[np.arange(10)] = [1, 2, 3]`)
- The iteration order over advanced indexes used to be always C-order. In NumPy 1.9. the iteration order adapts to the inputs and is not guaranteed (with the exception of a *single* advanced index which is never reversed for compatibility reasons). This means that the result is undefined if multiple values are assigned to the same element. An example for this is `arr[[0, 0], [1, 1]] = [1, 2]`, which may set `arr[0, 1]` to either 1 or 2.
- Equivalent to the iteration order, the memory layout of the advanced indexing result is adapted for faster indexing and cannot be predicted.
- All indexing operations return a view or a copy. No indexing operation will return the original array object. (For example `arr[...]`)
- In the future Boolean array-likes (such as lists of python bools) will always be treated as Boolean indexes and Boolean scalars (including python `True`) will be a legal *boolean* index. At this time, this is already the case for scalar arrays to allow the general `positive = a[a > 0]` to work when `a` is zero dimensional.

- In NumPy 1.8 it was possible to use `array(True)` and `array(False)` equivalent to 1 and 0 if the result of the operation was a scalar. This will raise an error in NumPy 1.9 and, as noted above, treated as a boolean index in the future.
- All non-integer array-likes are deprecated, object arrays of custom integer like objects may have to be cast explicitly.
- The error reporting for advanced indexing is more informative, however the error type has changed in some cases. (Broadcasting errors of indexing arrays are reported as `IndexError`)
- Indexing with more then one ellipsis (`...`) is deprecated.

Non-integer reduction axis indexes are deprecated

Non-integer axis indexes to reduction ufuncs like *add.reduce* or *sum* are deprecated.

`promote_types` and string dtype

`promote_types` function now returns a valid string length when given an integer or float dtype as one argument and a string dtype as another argument. Previously it always returned the input string dtype, even if it wasn't long enough to store the max integer/float value converted to a string.

`can_cast` and string dtype

`can_cast` function now returns False in "safe" casting mode for integer/float dtype and string dtype if the string dtype length is not long enough to store the max integer/float value converted to a string. Previously `can_cast` in "safe" mode returned True for integer/float dtype and a string dtype of any length.

astype and string dtype

The `astype` method now returns an error if the string dtype to cast to is not long enough in "safe" casting mode to hold the max value of integer/float array that is being casted. Previously the casting was allowed even if the result was truncated.

npyio.recfromcsv keyword arguments change

npyio.recfromcsv no longer accepts the undocumented *update* keyword, which used to override the *dtype* keyword.

The `doc/swig` directory moved

The `doc/swig` directory has been moved to `tools/swig`.

The `npy_3kcompat.h` header changed

The unused `simple_capsule_dtor` function has been removed from `npy_3kcompat.h`. Note that this header is not meant to be used outside of numpy; other projects should be using their own copy of this file when needed.

Negative indices in C-Api `sq_item` and `sq_ass_item` sequence methods

When directly accessing the `sq_item` or `sq_ass_item` PyObject slots for item getting, negative indices will not be supported anymore. `PySequence_GetItem` and `PySequence_SetItem` however fix negative indices so that they can be used there.

NDIter

When `NpyIter_RemoveAxis` is now called, the iterator range will be reset.

When a multi index is being tracked and an iterator is not buffered, it is possible to use `NpyIter_RemoveAxis`. In this case an iterator can shrink in size. Because the total size of an iterator is limited, the iterator may be too large before these calls. In this case its size will be set to `-1` and an error issued not at construction time but when removing the multi index, setting the iterator range, or getting the next function.

This has no effect on currently working code, but highlights the necessity of checking for an error return if these conditions can occur. In most cases the arrays being iterated are as large as the iterator so that such a problem cannot occur.

This change was already applied to the 1.8.1 release.

`zeros_like` for string dtypes now returns empty strings

To match the *zeros* function *zeros_like* now returns an array initialized with empty strings instead of an array filled with *'0'*.

15.72.5 New Features

Percentile supports more interpolation options

`np.percentile` now has the interpolation keyword argument to specify in which way points should be interpolated if the percentiles fall between two values. See the documentation for the available options.

Generalized axis support for median and percentile

`np.median` and `np.percentile` now support generalized axis arguments like ufunc reductions do since 1.7. One can now say axis=(index, index) to pick a list of axes for the reduction. The `keepdims` keyword argument was also added to allow convenient broadcasting to arrays of the original shape.

Dtype parameter added to `np.linspace` and `np.logspace`

The returned data type from the `linspace` and `logspace` functions can now be specified using the dtype parameter.

More general `np.triu` and `np.tril` broadcasting

For arrays with `ndim` exceeding 2, these functions will now apply to the final two axes instead of raising an exception.

`tobytes` alias for `tostring` method

`ndarray.tobytes` and `MaskedArray.tobytes` have been added as aliases for `tostring` which exports arrays as `bytes`. This is more consistent in Python 3 where `str` and `bytes` are not the same.

Build system

Added experimental support for the ppc64le and OpenRISC architecture.

Compatibility to python `numbers` module

All numerical numpy types are now registered with the type hierarchy in the python `numbers` module.

`increasing` parameter added to `np.vander`

The ordering of the columns of the Vandermonde matrix can be specified with this new boolean argument.

`unique_counts` parameter added to `np.unique`

The number of times each unique item comes up in the input can now be obtained as an optional return value.

Support for median and percentile in nanfunctions

The `np.nanmedian` and `np.nanpercentile` functions behave like the median and percentile functions except that NaNs are ignored.

NumpyVersion class added

The class may be imported from numpy.lib and can be used for version comparison when the numpy version goes to 1.10.devel. For example:

```
>>> from numpy.lib import NumpyVersion
>>> if NumpyVersion(np.__version__) < '1.10.0':
...     print('Wow, that is an old NumPy version!')
```

Allow saving arrays with large number of named columns

The numpy storage format 1.0 only allowed the array header to have a total size of 65535 bytes. This can be exceeded by structured arrays with a large number of columns. A new format 2.0 has been added which extends the header size to 4 GiB. *np.save* will automatically save in 2.0 format if the data requires it, else it will always use the more compatible 1.0 format.

Full broadcasting support for `np.cross`

`np.cross` now properly broadcasts its two input arrays, even if they have different number of dimensions. In earlier versions this would result in either an error being raised, or wrong results computed.

15.72.6 Improvements

Better numerical stability for sum in some cases

Pairwise summation is now used in the sum method, but only along the fast axis and for groups of the values <= 8192 in length. This should also improve the accuracy of var and std in some common cases.

Percentile implemented in terms of `np.partition`

`np.percentile` has been implemented in terms of `np.partition` which only partially sorts the data via a selection algorithm. This improves the time complexity from `O(nlog(n))` to `O(n)`.

Performance improvement for `np.array`

The performance of converting lists containing arrays to arrays using `np.array` has been improved. It is now equivalent in speed to `np.vstack(list)`.

Performance improvement for `np.searchsorted`

For the built-in numeric types, `np.searchsorted` no longer relies on the data type's `compare` function to perform the search, but is now implemented by type specific functions. Depending on the size of the inputs, this can result in performance improvements over 2x.

Optional reduced verbosity for np.distutils

Set `numpy.distutils.system_info.system_info.verbosity = 0` and then calls to `numpy.distutils.system_info.get_info('blas_opt')` will not print anything on the output. This is mostly for other packages using numpy.distutils.

Covariance check in `np.random.multivariate_normal`

A `RuntimeWarning` warning is raised when the covariance matrix is not positive-semidefinite.

Polynomial Classes no longer template based

The polynomial classes have been refactored to use an abstract base class rather than a template in order to implement a common interface. This makes importing the polynomial package faster as the classes do not need to be compiled on import.

More GIL releases

Several more functions now release the Global Interpreter Lock allowing more efficient parallelization using the `threading` module. Most notably the GIL is now released for fancy indexing, `np.where` and the `random` module now uses a per-state lock instead of the GIL.

MaskedArray support for more complicated base classes

Built-in assumptions that the baseclass behaved like a plain array are being removed. In particular, `repr` and `str` should now work more reliably.

C-API

15.72.7 Deprecations

Non-integer scalars for sequence repetition

Using non-integer numpy scalars to repeat python sequences is deprecated. For example `np.float_(2) * [1]` will be an error in the future.

`select` input deprecations

The integer and empty input to `select` is deprecated. In the future only boolean arrays will be valid conditions and an empty `condlist` will be considered an input error instead of returning the default.

`rank` function

The `rank` function has been deprecated to avoid confusion with `numpy.linalg.matrix_rank`.

Object array equality comparisons

In the future object array comparisons both == and *np.equal* will not make use of identity checks anymore. For example:

```
>>> a = np.array([np.array([1, 2, 3]), 1])
>>> b = np.array([np.array([1, 2, 3]), 1])
>>> a == b
```

will consistently return False (and in the future an error) even if the array in *a* and *b* was the same object.

The equality operator == will in the future raise errors like *np.equal* if broadcasting or element comparisons, etc. fails.

Comparison with *arr == None* will in the future do an elementwise comparison instead of just returning False. Code should be using *arr is None*.

All of these changes will give Deprecation- or FutureWarnings at this time.

C-API

The utility function npy_PyFile_Dup and npy_PyFile_DupClose are broken by the internal buffering python 3 applies to its file objects. To fix this two new functions npy_PyFile_Dup2 and npy_PyFile_DupClose2 are declared in npy_3kcompat.h and the old functions are deprecated. Due to the fragile nature of these functions it is recommended to instead use the python API when possible.

This change was already applied to the 1.8.1 release.

15.73 NumPy 1.8.2 Release Notes

This is a bugfix only release in the 1.8.x series.

15.73.1 Issues fixed

- gh-4836: partition produces wrong results for multiple selections in equal ranges
- gh-4656: Make fftpack._raw_fft threadsafe
- gh-4628: incorrect argument order to _copyto in in np.nanmax, np.nanmin
- gh-4642: Hold GIL for converting dtypes types with fields
- gh-4733: fix np.linalg.svd(b, compute_uv=False)
- gh-4853: avoid unaligned simd load on reductions on i386
- gh-4722: Fix seg fault converting empty string to object
- gh-4613: Fix lack of NULL check in array_richcompare
- gh-4774: avoid unaligned access for strided byteswap
- gh-650: Prevent division by zero when creating arrays from some buffers
- gh-4602: ifort has issues with optimization flag O2, use O1

15.74 NumPy 1.8.1 Release Notes

This is a bugfix only release in the 1.8.x series.

15.74.1 Issues fixed

- gh-4276: Fix mean, var, std methods for object arrays
- gh-4262: remove insecure mktemp usage
- gh-2385: absolute(complex(inf)) raises invalid warning in python3
- gh-4024: Sequence assignment doesn't raise exception on shape mismatch
- gh-4027: Fix chunked reading of strings longer than BUFFERSIZE
- gh-4109: Fix object scalar return type of 0-d array indices
- gh-4018: fix missing check for memory allocation failure in ufuncs
- gh-4156: high order linalg.norm discards imaginary elements of complex arrays
- gh-4144: linalg: norm fails on longdouble, signed int
- gh-4094: fix NaT handling in _strided_to_strided_string_to_datetime
- gh-4051: fix uninitialized use in _strided_to_strided_string_to_datetime
- gh-4093: Loading compressed .npz file fails under Python 2.6.6
- gh-4138: segfault with non-native endian memoryview in python 3.4
- gh-4123: Fix missing NULL check in lexsort
- gh-4170: fix native-only long long check in memoryviews
- gh-4187: Fix large file support on 32 bit
- gh-4152: fromfile: ensure file handle positions are in sync in python3
- gh-4176: clang compatibility: Typos in conversion_utils
- gh-4223: Fetching a non-integer item caused array return
- gh-4197: fix minor memory leak in memoryview failure case
- gh-4206: fix build with single-threaded python
- gh-4220: add versionadded:: 1.8.0 to ufunc.at docstring
- gh-4267: improve handling of memory allocation failure
- gh-4267: fix use of capi without gil in ufunc.at
- gh-4261: Detect vendor versions of GNU Compilers
- gh-4253: IRR was returning nan instead of valid negative answer
- gh-4254: fix unnecessary byte order flag change for byte arrays
- gh-3263: numpy.random.shuffle clobbers mask of a MaskedArray
- gh-4270: np.random.shuffle not work with flexible dtypes
- gh-3173: Segmentation fault when 'size' argument to random.multinomial
- gh-2799: allow using unique with lists of complex

- gh-3504: fix linspace truncation for integer array scalar
- gh-4191: get_info('openblas') does not read libraries key
- gh-3348: Access violation in _descriptor_from_pep3118_format
- gh-3175: segmentation fault with numpy.array() from bytearray
- gh-4266: histogramdd - wrong result for entries very close to last boundary
- gh-4408: Fix stride_stricks.as_strided function for object arrays
- gh-4225: fix log1p and exmp1 return for np.inf on windows compiler builds
- gh-4359: Fix infinite recursion in str.format of flex arrays
- gh-4145: Incorrect shape of broadcast result with the exponent operator
- gh-4483: Fix commutativity of {dot,multiply,inner}(scalar, matrix_of_objs)
- gh-4466: Delay npyiter size check when size may change
- gh-4485: Buffered stride was erroneously marked fixed
- gh-4354: byte_bounds fails with datetime dtypes
- gh-4486: segfault/error converting from/to high-precision datetime64 objects
- gh-4428: einsum(None, None, None, None) causes segfault
- gh-4134: uninitialized use for for size 1 object reductions

15.74.2 Changes

NDIter

When `NpyIter_RemoveAxis` is now called, the iterator range will be reset.

When a multi index is being tracked and an iterator is not buffered, it is possible to use `NpyIter_RemoveAxis`. In this case an iterator can shrink in size. Because the total size of an iterator is limited, the iterator may be too large before these calls. In this case its size will be set to `-1` and an error issued not at construction time but when removing the multi index, setting the iterator range, or getting the next function.

This has no effect on currently working code, but highlights the necessity of checking for an error return if these conditions can occur. In most cases the arrays being iterated are as large as the iterator so that such a problem cannot occur.

Optional reduced verbosity for np.distutils

Set `numpy.distutils.system_info.system_info.verbosity = 0` and then calls to `numpy.distutils.system_info.get_info('blas_opt')` will not print anything on the output. This is mostly for other packages using numpy.distutils.

15.74.3 Deprecations

C-API

The utility function npy_PyFile_Dup and npy_PyFile_DupClose are broken by the internal buffering python 3 applies to its file objects. To fix this two new functions npy_PyFile_Dup2 and npy_PyFile_DupClose2 are declared in npy_3kcompat.h and the old functions are deprecated. Due to the fragile nature of these functions it is recommended to instead use the python API when possible.

15.75 NumPy 1.8.0 Release Notes

This release supports Python 2.6 -2.7 and 3.2 - 3.3.

15.75.1 Highlights

- New, no 2to3, Python 2 and Python 3 are supported by a common code base.
- New, gufuncs for linear algebra, enabling operations on stacked arrays.
- New, inplace fancy indexing for ufuncs with the `.at` method.
- New, `partition` function, partial sorting via selection for fast median.
- New, `nanmean`, `nanvar`, and `nanstd` functions skipping NaNs.
- New, `full` and `full_like` functions to create value initialized arrays.
- New, `PyUFunc_RegisterLoopForDescr`, better ufunc support for user dtypes.
- Numerous performance improvements in many areas.

15.75.2 Dropped Support

Support for Python versions 2.4 and 2.5 has been dropped,

Support for SCons has been removed.

15.75.3 Future Changes

The Datetime64 type remains experimental in this release. In 1.9 there will probably be some changes to make it more usable.

The diagonal method currently returns a new array and raises a FutureWarning. In 1.9 it will return a readonly view.

Multiple field selection from an array of structured type currently returns a new array and raises a FutureWarning. In 1.9 it will return a readonly view.

The numpy/oldnumeric and numpy/numarray compatibility modules will be removed in 1.9.

15.75.4 Compatibility notes

The doc/sphinxext content has been moved into its own github repository, and is included in numpy as a submodule. See the instructions in doc/HOWTO_BUILD_DOCS.rst.txt for how to access the content.

The hash function of numpy.void scalars has been changed. Previously the pointer to the data was hashed as an integer. Now, the hash function uses the tuple-hash algorithm to combine the hash functions of the elements of the scalar, but only if the scalar is read-only.

Numpy has switched its build system to using 'separate compilation' by default. In previous releases this was supported, but not default. This should produce the same results as the old system, but if you're trying to do something complicated like link numpy statically or using an unusual compiler, then it's possible you will encounter problems. If so, please file a bug and as a temporary workaround you can re-enable the old build system by exporting the shell variable NPY_SEPARATE_COMPILATION=0.

For the AdvancedNew iterator the `oa_ndim` flag should now be -1 to indicate that no `op_axes` and `itershape` are passed in. The `oa_ndim == 0` case, now indicates a 0-D iteration and `op_axes` being NULL and the old usage is deprecated. This does not effect the `NpyIter_New` or `NpyIter_MultiNew` functions.

The functions nanargmin and nanargmax now return np.iinfo['intp'].min for the index in all-NaN slices. Previously the functions would raise a ValueError for array returns and NaN for scalar returns.

NPY_RELAXED_STRIDES_CHECKING

There is a new compile time environment variable `NPY_RELAXED_STRIDES_CHECKING`. If this variable is set to 1, then numpy will consider more arrays to be C- or F-contiguous – for example, it becomes possible to have a column vector which is considered both C- and F-contiguous simultaneously. The new definition is more accurate, allows for faster code that makes fewer unnecessary copies, and simplifies numpy's code internally. However, it may also break third-party libraries that make too-strong assumptions about the stride values of C- and F-contiguous arrays. (It is also currently known that this breaks Cython code using memoryviews, which will be fixed in Cython.) THIS WILL BECOME THE DEFAULT IN A FUTURE RELEASE, SO PLEASE TEST YOUR CODE NOW AGAINST NUMPY BUILT WITH:

```
NPY_RELAXED_STRIDES_CHECKING=1 python setup.py install
```

You can check whether NPY_RELAXED_STRIDES_CHECKING is in effect by running:

```
np.ones((10, 1), order="C").flags.f_contiguous
```

This will be `True` if relaxed strides checking is enabled, and `False` otherwise. The typical problem we've seen so far is C code that works with C-contiguous arrays, and assumes that the itemsize can be accessed by looking at the last element in the `PyArray_STRIDES(arr)` array. When relaxed strides are in effect, this is not true (and in fact, it never was true in some corner cases). Instead, use `PyArray_ITEMSIZE(arr)`.

For more information check the "Internal memory layout of an ndarray" section in the documentation.

Binary operations with non-arrays as second argument

Binary operations of the form `<array-or-subclass> * <non-array-subclass>` where `<non-array-subclass>` declares an `__array_priority__` higher than that of `<array-or-subclass>` will now unconditionally return *NotImplemented*, giving `<non-array-subclass>` a chance to handle the operation. Previously, *NotImplemented* would only be returned if `<non-array-subclass>` actually implemented the reversed operation, and after a (potentially expensive) array conversion of `<non-array-subclass>` had been attempted. (bug, pull request)

Function *median* used with *overwrite_input* only partially sorts array

If *median* is used with *overwrite_input* option the input array will now only be partially sorted instead of fully sorted.

Fix to financial.npv

The npv function had a bug. Contrary to what the documentation stated, it summed from indexes `1` to `M` instead of from `0` to `M - 1`. The fix changes the returned value. The mirr function called the npv function, but worked around the problem, so that was also fixed and the return value of the mirr function remains unchanged.

Runtime warnings when comparing NaN numbers

Comparing `NaN` floating point numbers now raises the `invalid` runtime warning. If a `NaN` is expected the warning can be ignored using np.errstate. E.g.:

```
with np.errstate(invalid='ignore'):
    operation()
```

15.75.5 New Features

Support for linear algebra on stacked arrays

The gufunc machinery is now used for np.linalg, allowing operations on stacked arrays and vectors. For example:

```
>>> a
array([[[ 1.,  1.],
        [ 0.,  1.]],

       [[ 1.,  1.],
        [ 0.,  1.]]])

>>> np.linalg.inv(a)
array([[[ 1., -1.],
        [ 0.,  1.]],

       [[ 1., -1.],
        [ 0.,  1.]]])
```

In place fancy indexing for ufuncs

The function `at` has been added to ufunc objects to allow in place ufuncs with no buffering when fancy indexing is used. For example, the following will increment the first and second items in the array, and will increment the third item twice: `numpy.add.at(arr, [0, 1, 2, 2], 1)`

This is what many have mistakenly thought `arr[[0, 1, 2, 2]] += 1` would do, but that does not work as the incremented value of `arr[2]` is simply copied into the third slot in `arr` twice, not incremented twice.

New functions *partition* and *argpartition*

New functions to partially sort arrays via a selection algorithm.

A `partition` by index `k` moves the `k` smallest element to the front of an array. All elements before `k` are then smaller or equal than the value in position `k` and all elements following `k` are then greater or equal than the value in position `k`. The ordering of the values within these bounds is undefined. A sequence of indices can be provided to sort all of them into their sorted position at once iterative partitioning. This can be used to efficiently obtain order statistics like median or percentiles of samples. `partition` has a linear time complexity of `O(n)` while a full sort has `O(n log(n))`.

New functions *nanmean*, *nanvar* and *nanstd*

New nan aware statistical functions are added. In these functions the results are what would be obtained if nan values were omitted from all computations.

New functions *full* and *full_like*

New convenience functions to create arrays filled with a specific value; complementary to the existing *zeros* and *zeros_like* functions.

IO compatibility with large files

Large NPZ files >2GB can be loaded on 64-bit systems.

Building against OpenBLAS

It is now possible to build numpy against OpenBLAS by editing site.cfg.

New constant

Euler's constant is now exposed in numpy as euler_gamma.

New modes for qr

New modes 'complete', 'reduced', and 'raw' have been added to the qr factorization and the old 'full' and 'economic' modes are deprecated. The 'reduced' mode replaces the old 'full' mode and is the default as was the 'full' mode, so backward compatibility can be maintained by not specifying the mode.

The 'complete' mode returns a full dimensional factorization, which can be useful for obtaining a basis for the orthogonal complement of the range space. The 'raw' mode returns arrays that contain the Householder reflectors and scaling factors that can be used in the future to apply q without needing to convert to a matrix. The 'economic' mode is simply deprecated, there isn't much use for it and it isn't any more efficient than the 'raw' mode.

New *invert* argument to *in1d*

The function *in1d* now accepts a *invert* argument which, when *True*, causes the returned array to be inverted.

Advanced indexing using *np.newaxis*

It is now possible to use *np.newaxis*/*None* together with index arrays instead of only in simple indices. This means that `array[np.newaxis, [0, 1]]` will now work as expected and select the first two rows while prepending a new axis to the array.

C-API

New ufuncs can now be registered with builtin input types and a custom output type. Before this change, NumPy wouldn't be able to find the right ufunc loop function when the ufunc was called from Python, because the ufunc loop signature matching logic wasn't looking at the output operand type. Now the correct ufunc loop is found, as long as the user provides an output argument with the correct output type.

runtests.py

A simple test runner script `runtests.py` was added. It also builds Numpy via `setup.py build` and can be used to run tests easily during development.

15.75.6 Improvements

IO performance improvements

Performance in reading large files was improved by chunking (see also IO compatibility).

Performance improvements to *pad*

The *pad* function has a new implementation, greatly improving performance for all inputs except *mode*= (retained for backwards compatibility). Scaling with dimensionality is dramatically improved for rank >= 4.

Performance improvements to *isnan*, *isinf*, *isfinite* and *byteswap*

isnan, *isinf*, *isfinite* and *byteswap* have been improved to take advantage of compiler builtins to avoid expensive calls to libc. This improves performance of these operations by about a factor of two on gnu libc systems.

Performance improvements via SSE2 vectorization

Several functions have been optimized to make use of SSE2 CPU SIMD instructions.

- **Float32 and float64:**
  - base math (*add*, *subtract*, *divide*, *multiply*)
  - *sqrt*
  - *minimum/maximum*
  - *absolute*

- **Bool:**
  - *logical_or*
  - *logical_and*
  - *logical_not*

This improves performance of these operations up to 4x/2x for float32/float64 and up to 10x for bool depending on the location of the data in the CPU caches. The performance gain is greatest for in-place operations.

In order to use the improved functions the SSE2 instruction set must be enabled at compile time. It is enabled by default on x86_64 systems. On x86_32 with a capable CPU it must be enabled by passing the appropriate flag to the CFLAGS build variable (-msse2 with gcc).

Performance improvements to *median*

median is now implemented in terms of *partition* instead of *sort* which reduces its time complexity from O(n log(n)) to O(n). If used with the *overwrite_input* option the array will now only be partially sorted instead of fully sorted.

Overridable operand flags in ufunc C-API

When creating a ufunc, the default ufunc operand flags can be overridden via the new op_flags attribute of the ufunc object. For example, to set the operand flag for the first input to read/write:

PyObject *ufunc = PyUFunc_FromFuncAndData(…); ufunc->op_flags[0] = NPY_ITER_READWRITE;

This allows a ufunc to perform an operation in place. Also, global nditer flags can be overridden via the new iter_flags attribute of the ufunc object. For example, to set the reduce flag for a ufunc:

ufunc->iter_flags = NPY_ITER_REDUCE_OK;

15.75.7 Changes

General

The function np.take now allows 0-d arrays as indices.

The separate compilation mode is now enabled by default.

Several changes to np.insert and np.delete:

- Previously, negative indices and indices that pointed past the end of the array were simply ignored. Now, this will raise a Future or Deprecation Warning. In the future they will be treated like normal indexing treats them – negative indices will wrap around, and out-of-bound indices will generate an error.
- Previously, boolean indices were treated as if they were integers (always referring to either the 0th or 1st item in the array). In the future, they will be treated as masks. In this release, they raise a FutureWarning warning of this coming change.
- In Numpy 1.7. np.insert already allowed the syntax *np.insert(arr, 3, [1,2,3])* to insert multiple items at a single position. In Numpy 1.8. this is also possible for *np.insert(arr, [3], [1, 2, 3])*.

Padded regions from np.pad are now correctly rounded, not truncated.

C-API Array Additions

Four new functions have been added to the array C-API.

- PyArray_Partition
- PyArray_ArgPartition
- PyArray_SelectkindConverter
- PyDataMem_NEW_ZEROED

C-API Ufunc Additions

One new function has been added to the ufunc C-API that allows to register an inner loop for user types using the descr.

- PyUFunc_RegisterLoopForDescr

C-API Developer Improvements

The `PyArray_Type` instance creation function `tp_new` now uses `tp_basicsize` to determine how much memory to allocate. In previous releases only `sizeof(PyArrayObject)` bytes of memory were allocated, often requiring C-API subtypes to reimplement `tp_new`.

15.75.8 Deprecations

The 'full' and 'economic' modes of qr factorization are deprecated.

General

The use of non-integer for indices and most integer arguments has been deprecated. Previously float indices and function arguments such as axes or shapes were truncated to integers without warning. For example *arr.reshape(3., -1)* or *arr[0.]* will trigger a deprecation warning in NumPy 1.8., and in some future version of NumPy they will raise an error.

15.75.9 Authors

This release contains work by the following people who contributed at least one patch to this release. The names are in alphabetical order by first name:

- 87
- Adam Ginsburg +
- Adam Griffiths +
- Alexander Belopolsky +
- Alex Barth +
- Alex Ford +
- Andreas Hilboll +
- Andreas Kloeckner +
- Andreas Schwab +

- Andrew Horton +
- argriffing +
- Arink Verma +
- Bago Amirbekian +
- Bartosz Telenczuk +
- bebert218 +
- Benjamin Root +
- Bill Spotz +
- Bradley M. Froehle
- Carwyn Pelley +
- Charles Harris
- Chris
- Christian Brueffer +
- Christoph Dann +
- Christoph Gohlke
- Dan Hipschman +
- Daniel +
- Dan Miller +
- daveydave400 +
- David Cournapeau
- David Warde-Farley
- Denis Laxalde
- dmuellner +
- Edward Catmur +
- Egor Zindy +
- endolith
- Eric Firing
- Eric Fode
- Eric Moore +
- Eric Price +
- Fazlul Shahriar +
- Félix Hartmann +
- Fernando Perez
- Frank B +
- Frank Breitling +
- Frederic

- Gabriel
- GaelVaroquaux
- Guillaume Gay +
- Han Genuit
- HaroldMills +
- hklemm +
- jamestwebber +
- Jason Madden +
- Jay Bourque
- jeromekelleher +
- Jesús Gómez +
- jmozmoz +
- jnothman +
- Johannes Schönberger +
- John Benediktsson +
- John Salvatier +
- John Stechschulte +
- Jonathan Waltman +
- Joon Ro +
- Jos de Kloe +
- Joseph Martinot-Lagarde +
- Josh Warner (Mac) +
- Jostein Bø Fløystad +
- Juan Luis Cano Rodríguez +
- Julian Taylor +
- Julien Phalip +
- K.-Michael Aye +
- Kumar Appaiah +
- Lars Buitinck
- Leon Weber +
- Luis Pedro Coelho
- Marcin Juszkiewicz
- Mark Wiebe
- Marten van Kerkwijk +
- Martin Baeuml +
- Martin Spacek

- Martin Teichmann +
- Matt Davis +
- Matthew Brett
- Maximilian Albert +
- m-d-w +
- Michael Droettboom
- mwtoews +
- Nathaniel J. Smith
- Nicolas Scheffer +
- Nils Werner +
- ochoadavid +
- Ondřej Čertík
- ovillellas +
- Paul Ivanov
- Pauli Virtanen
- peterjc
- Ralf Gommers
- Raul Cota +
- Richard Hattersley +
- Robert Costa +
- Robert Kern
- Rob Ruana +
- Ronan Lamy
- Sandro Tosi
- Sascha Peilicke +
- Sebastian Berg
- Skipper Seabold
- Stefan van der Walt
- Steve +
- Takafumi Arakaki +
- Thomas Robitaille +
- Tomas Tomecek +
- Travis E. Oliphant
- Valentin Haenel
- Vladimir Rutsky +
- Warren Weckesser

- Yaroslav Halchenko
- Yury V. Zaytsev +

A total of 119 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

15.76 NumPy 1.7.2 Release Notes

This is a bugfix only release in the 1.7.x series. It supports Python 2.4 - 2.7 and 3.1 - 3.3 and is the last series that supports Python 2.4 - 2.5.

15.76.1 Issues fixed

- gh-3153: Do not reuse nditer buffers when not filled enough
- gh-3192: f2py crashes with UnboundLocalError exception
- gh-442: Concatenate with axis=None now requires equal number of array elements
- gh-2485: Fix for astype('S') string truncate issue
- gh-3312: bug in count_nonzero
- gh-2684: numpy.ma.average casts complex to float under certain conditions
- gh-2403: masked array with named components does not behave as expected
- gh-2495: np.ma.compress treated inputs in wrong order
- gh-576: add __len__ method to ma.mvoid
- gh-3364: reduce performance regression of mmap slicing
- gh-3421: fix non-swapping strided copies in GetStridedCopySwap
- gh-3373: fix small leak in datetime metadata initialization
- gh-2791: add platform specific python include directories to search paths
- gh-3168: fix undefined function and add integer divisions
- gh-3301: memmap does not work with TemporaryFile in python3
- gh-3057: distutils.misc_util.get_shared_lib_extension returns wrong debug extension
- gh-3472: add module extensions to load_library search list
- gh-3324: Make comparison function (gt, ge, …) respect __array_priority__
- gh-3497: np.insert behaves incorrectly with argument 'axis=-1'
- gh-3541: make preprocessor tests consistent in halffloat.c
- gh-3458: array_ass_boolean_subscript() writes 'non-existent' data to array
- gh-2892: Regression in ufunc.reduceat with zero-sized index array
- gh-3608: Regression when filling struct from tuple
- gh-3701: add support for Python 3.4 ast.NameConstant
- gh-3712: do not assume that GIL is enabled in xerbla
- gh-3712: fix LAPACK error handling in lapack_litemodule

- gh-3728: f2py fix decref on wrong object
- gh-3743: Hash changed signature in Python 3.3
- gh-3793: scalar int hashing broken on 64 bit python3
- gh-3160: SandboxViolation easyinstalling 1.7.0 on Mac OS X 10.8.3
- gh-3871: npy_math.h has invalid isinf for Solaris with SUNWspro12.2
- gh-2561: Disable check for oldstyle classes in python3
- gh-3900: Ensure NotImplemented is passed on in MaskedArray ufunc's
- gh-2052: del scalar subscript causes segfault
- gh-3832: fix a few uninitialized uses and memleaks
- gh-3971: f2py changed string.lowercase to string.ascii_lowercase for python3
- gh-3480: numpy.random.binomial raised ValueError for n == 0
- gh-3992: hypot(inf, 0) shouldn't raise a warning, hypot(inf, inf) wrong result
- gh-4018: Segmentation fault dealing with very large arrays
- gh-4094: fix NaT handling in _strided_to_strided_string_to_datetime
- gh-4051: fix uninitialized use in _strided_to_strided_string_to_datetime
- gh-4123: lexsort segfault
- gh-4141: Fix a few issues that show up with python 3.4b1

15.77 NumPy 1.7.1 Release Notes

This is a bugfix only release in the 1.7.x series. It supports Python 2.4 - 2.7 and 3.1 - 3.3 and is the last series that supports Python 2.4 - 2.5.

15.77.1 Issues fixed

- gh-2973: Fix *1* is printed during numpy.test()
- gh-2983: BUG: gh-2969: Backport memory leak fix 80b3a34.
- gh-3007: Backport gh-3006
- gh-2984: Backport fix complex polynomial fit
- gh-2982: BUG: Make nansum work with booleans.
- gh-2985: Backport large sort fixes
- gh-3039: Backport object take
- gh-3105: Backport nditer fix op axes initialization
- gh-3108: BUG: npy-pkg-config ini files were missing after Bento build.
- gh-3124: BUG: PyArray_LexSort allocates too much temporary memory.
- gh-3131: BUG: Exported f2py_size symbol prevents linking multiple f2py modules.
- gh-3117: Backport gh-2992

- gh-3135: DOC: Add mention of PyArray_SetBaseObject stealing a reference
- gh-3134: DOC: Fix typo in fft docs (the indexing variable is 'm', not 'n').
- gh-3136: Backport #3128

15.78 NumPy 1.7.0 Release Notes

This release includes several new features as well as numerous bug fixes and refactorings. It supports Python 2.4 - 2.7 and 3.1 - 3.3 and is the last release that supports Python 2.4 - 2.5.

15.78.1 Highlights

- `where=` parameter to ufuncs (allows the use of boolean arrays to choose where a computation should be done)
- `vectorize` improvements (added 'excluded' and 'cache' keyword, general cleanup and bug fixes)
- `numpy.random.choice` (random sample generating function)

15.78.2 Compatibility notes

In a future version of numpy, the functions np.diag, np.diagonal, and the diagonal method of ndarrays will return a view onto the original array, instead of producing a copy as they do now. This makes a difference if you write to the array returned by any of these functions. To facilitate this transition, numpy 1.7 produces a FutureWarning if it detects that you may be attempting to write to such an array. See the documentation for np.diagonal for details.

Similar to np.diagonal above, in a future version of numpy, indexing a record array by a list of field names will return a view onto the original array, instead of producing a copy as they do now. As with np.diagonal, numpy 1.7 produces a FutureWarning if it detects that you may be attempting to write to such an array. See the documentation for array indexing for details.

In a future version of numpy, the default casting rule for UFunc out= parameters will be changed from 'unsafe' to 'same_kind'. (This also applies to in-place operations like a += b, which is equivalent to np.add(a, b, out=a).) Most usages which violate the 'same_kind' rule are likely bugs, so this change may expose previously undetected errors in projects that depend on NumPy. In this version of numpy, such usages will continue to succeed, but will raise a DeprecationWarning.

Full-array boolean indexing has been optimized to use a different, optimized code path. This code path should produce the same results, but any feedback about changes to your code would be appreciated.

Attempting to write to a read-only array (one with `arr.flags.writeable` set to `False`) used to raise either a RuntimeError, ValueError, or TypeError inconsistently, depending on which code path was taken. It now consistently raises a ValueError.

The <ufunc>.reduce functions evaluate some reductions in a different order than in previous versions of NumPy, generally providing higher performance. Because of the nature of floating-point arithmetic, this may subtly change some results, just as linking NumPy to a different BLAS implementations such as MKL can.

If upgrading from 1.5, then generally in 1.6 and 1.7 there have been substantial code added and some code paths altered, particularly in the areas of type resolution and buffered iteration over universal functions. This might have an impact on your code particularly if you relied on accidental behavior in the past.

15.78.3 New features

Reduction UFuncs Generalize axis= Parameter

Any ufunc.reduce function call, as well as other reductions like sum, prod, any, all, max and min support the ability to choose a subset of the axes to reduce over. Previously, one could say axis=None to mean all the axes or axis=# to pick a single axis. Now, one can also say axis=(#,#) to pick a list of axes for reduction.

Reduction UFuncs New keepdims= Parameter

There is a new keepdims= parameter, which if set to True, doesn't throw away the reduction axes but instead sets them to have size one. When this option is set, the reduction result will broadcast correctly to the original operand which was reduced.

Datetime support

Note: The datetime API is *experimental* in 1.7.0, and may undergo changes in future versions of NumPy.

There have been a lot of fixes and enhancements to datetime64 compared to NumPy 1.6:

- the parser is quite strict about only accepting ISO 8601 dates, with a few convenience extensions
- converts between units correctly
- datetime arithmetic works correctly
- business day functionality (allows the datetime to be used in contexts where only certain days of the week are valid)

The notes in doc/source/reference/arrays.datetime.rst (also available in the online docs at arrays.datetime.html) should be consulted for more details.

Custom formatter for printing arrays

See the new `formatter` parameter of the `numpy.set_printoptions` function.

New function numpy.random.choice

A generic sampling function has been added which will generate samples from a given array-like. The samples can be with or without replacement, and with uniform or given non-uniform probabilities.

New function isclose

Returns a boolean array where two arrays are element-wise equal within a tolerance. Both relative and absolute tolerance can be specified.

Preliminary multi-dimensional support in the polynomial package

Axis keywords have been added to the integration and differentiation functions and a tensor keyword was added to the evaluation functions. These additions allow multi-dimensional coefficient arrays to be used in those functions. New functions for evaluating 2-D and 3-D coefficient arrays on grids or sets of points were added together with 2-D and 3-D pseudo-Vandermonde matrices that can be used for fitting.

Ability to pad rank-n arrays

A pad module containing functions for padding n-dimensional arrays has been added. The various private padding functions are exposed as options to a public 'pad' function. Example:

```
pad(a, 5, mode='mean')
```

Current modes are `constant`, `edge`, `linear_ramp`, `maximum`, `mean`, `median`, `minimum`, `reflect`, `symmetric`, `wrap`, and `<function>`.

New argument to searchsorted

The function searchsorted now accepts a 'sorter' argument that is a permutation array that sorts the array to search.

Build system

Added experimental support for the AArch64 architecture.

C API

New function `PyArray_FailUnlessWriteable` provides a consistent interface for checking array writeability – any C code which works with arrays whose WRITEABLE flag is not known to be True a priori, should make sure to call this function before writing.

NumPy C Style Guide added (`doc/C_STYLE_GUIDE.rst.txt`).

15.78.4 Changes

General

The function np.concatenate tries to match the layout of its input arrays. Previously, the layout did not follow any particular reason, and depended in an undesirable way on the particular axis chosen for concatenation. A bug was also fixed which silently allowed out of bounds axis arguments.

The ufuncs logical_or, logical_and, and logical_not now follow Python's behavior with object arrays, instead of trying to call methods on the objects. For example the expression (3 and 'test') produces the string 'test', and now np.logical_and(np.array(3, 'O'), np.array('test', 'O')) produces 'test' as well.

The `.base` attribute on ndarrays, which is used on views to ensure that the underlying array owning the memory is not deallocated prematurely, now collapses out references when you have a view-of-a-view. For example:

```
a = np.arange(10)
b = a[1:]
c = b[1:]
```

In numpy 1.6, `c.base` is `b`, and `c.base.base` is `a`. In numpy 1.7, `c.base` is `a`.

To increase backwards compatibility for software which relies on the old behaviour of `.base`, we only 'skip over' objects which have exactly the same type as the newly created view. This makes a difference if you use `ndarray` subclasses. For example, if we have a mix of `ndarray` and `matrix` objects which are all views on the same original `ndarray`:

```
a = np.arange(10)
b = np.asmatrix(a)
c = b[0, 1:]
d = c[0, 1:]
```

then `d.base` will be `b`. This is because `d` is a `matrix` object, and so the collapsing process only continues so long as it encounters other `matrix` objects. It considers `c`, `b`, and `a` in that order, and `b` is the last entry in that list which is a `matrix` object.

Casting Rules

Casting rules have undergone some changes in corner cases, due to the NA-related work. In particular for combinations of scalar+scalar:

- the *longlong* type (*q*) now stays *longlong* for operations with any other number (*? b h i l q p B H I*), previously it was cast as *int_* (*l*). The *ulonglong* type (*Q*) now stays as *ulonglong* instead of *uint* (*L*).
- the *timedelta64* type (*m*) can now be mixed with any integer type (*b h i l q p B H I L Q P*), previously it raised *TypeError*.

For array + scalar, the above rules just broadcast except the case when the array and scalars are unsigned/signed integers, then the result gets converted to the array type (of possibly larger size) as illustrated by the following examples:

```
>>> (np.zeros((2,), dtype=np.uint8) + np.int16(257)).dtype
dtype('uint16')
>>> (np.zeros((2,), dtype=np.int8) + np.uint16(257)).dtype
dtype('int16')
>>> (np.zeros((2,), dtype=np.int16) + np.uint32(2**17)).dtype
dtype('int32')
```

Whether the size gets increased depends on the size of the scalar, for example:

```
>>> (np.zeros((2,), dtype=np.uint8) + np.int16(255)).dtype
dtype('uint8')
>>> (np.zeros((2,), dtype=np.uint8) + np.int16(256)).dtype
dtype('uint16')
```

Also a `complex128` scalar + `float32` array is cast to `complex64`.

In NumPy 1.7 the *datetime64* type (*M*) must be constructed by explicitly specifying the type as the second argument (e.g. `np.datetime64(2000, 'Y')`).

15.78.5 Deprecations

General

Specifying a custom string formatter with a *_format* array attribute is deprecated. The new *formatter* keyword in `numpy.set_printoptions` or `numpy.array2string` can be used instead.

The deprecated imports in the polynomial package have been removed.

`concatenate` now raises DepractionWarning for 1D arrays if `axis != 0`. Versions of numpy < 1.7.0 ignored axis argument value for 1D arrays. We allow this for now, but in due course we will raise an error.

C-API

Direct access to the fields of PyArrayObject* has been deprecated. Direct access has been recommended against for many releases. Expect similar deprecations for PyArray_Descr* and other core objects in the future as preparation for NumPy 2.0.

The macros in old_defines.h are deprecated and will be removed in the next major release (>= 2.0). The sed script tools/replace_old_macros.sed can be used to replace these macros with the newer versions.

You can test your code against the deprecated C API by adding a line composed of `#define NPY_NO_DEPRECATED_API` and the target version number, such as `NPY_1_7_API_VERSION`, before including any NumPy headers.

The `NPY_CHAR` member of the `NPY_TYPES` enum is deprecated and will be removed in NumPy 1.8. See the discussion at gh-2801 for more details.

15.79 NumPy 1.6.2 Release Notes

This is a bugfix release in the 1.6.x series. Due to the delay of the NumPy 1.7.0 release, this release contains far more fixes than a regular NumPy bugfix release. It also includes a number of documentation and build improvements.

15.79.1 Issues fixed

`numpy.core`

- #2063: make unique() return consistent index
- #1138: allow creating arrays from empty buffers or empty slices
- #1446: correct note about correspondence vstack and concatenate
- #1149: make argmin() work for datetime
- #1672: fix allclose() to work for scalar inf
- #1747: make np.median() work for 0-D arrays
- #1776: make complex division by zero to yield inf properly
- #1675: add scalar support for the format() function
- #1905: explicitly check for NaNs in allclose()
- #1952: allow floating ddof in std() and var()

- #1948: fix regression for indexing chararrays with empty list
- #2017: fix type hashing
- #2046: deleting array attributes causes segfault
- #2033: a**2.0 has incorrect type
- #2045: make attribute/iterator_element deletions not segfault
- #2021: fix segfault in searchsorted()
- #2073: fix float16 __array_interface__ bug

`numpy.lib`

- #2048: break reference cycle in NpzFile
- #1573: savetxt() now handles complex arrays
- #1387: allow bincount() to accept empty arrays
- #1899: fixed histogramdd() bug with empty inputs
- #1793: fix failing npyio test under py3k
- #1936: fix extra nesting for subarray dtypes
- #1848: make tril/triu return the same dtype as the original array
- #1918: use Py_TYPE to access ob_type, so it works also on Py3

`numpy.distutils`

- #1261: change compile flag on AIX from -O5 to -O3
- #1377: update HP compiler flags
- #1383: provide better support for C++ code on HPUX
- #1857: fix build for py3k + pip
- BLD: raise a clearer warning in case of building without cleaning up first
- BLD: follow build_ext coding convention in build_clib
- BLD: fix up detection of Intel CPU on OS X in system_info.py
- BLD: add support for the new X11 directory structure on Ubuntu & co.
- BLD: add support for the new X11 directory structure on Ubuntu & co.
- BLD: add ufsparse to the libraries search path.
- BLD: add 'pgfortran' as a valid compiler in the Portland Group
- BLD: update version match regexp for IBM AIX Fortran compilers.

numpy.random

- BUG: Use npy_intp instead of long in mtrand

15.79.2 Changes

numpy.f2py

- ENH: Introduce new options extra_f77_compiler_args and extra_f90_compiler_args
- BLD: Improve reporting of fcompiler value
- BUG: Fix f2py test_kind.py test

numpy.poly

- ENH: Add some tests for polynomial printing
- ENH: Add companion matrix functions
- DOC: Rearrange the polynomial documents
- BUG: Fix up links to classes
- DOC: Add version added to some of the polynomial package modules
- DOC: Document xxxfit functions in the polynomial package modules
- BUG: The polynomial convenience classes let different types interact
- DOC: Document the use of the polynomial convenience classes
- DOC: Improve numpy reference documentation of polynomial classes
- ENH: Improve the computation of polynomials from roots
- STY: Code cleanup in polynomial [*]fromroots functions
- DOC: Remove references to cast and NA, which were added in 1.7

15.80 NumPy 1.6.1 Release Notes

This is a bugfix only release in the 1.6.x series.

15.80.1 Issues Fixed

- #1834: einsum fails for specific shapes
- #1837: einsum throws nan or freezes python for specific array shapes
- #1838: object <-> structured type arrays regression
- #1851: regression for SWIG based code in 1.6.0
- #1863: Buggy results when operating on array copied with astype()
- #1870: Fix corner case of object array assignment
- #1843: Py3k: fix error with recarray

- #1885: nditer: Error in detecting double reduction loop
- #1874: f2py: fix –include_paths bug
- #1749: Fix ctypes.load_library()
- #1895/1896: iter: writeonly operands weren't always being buffered correctly

15.81 NumPy 1.6.0 Release Notes

This release includes several new features as well as numerous bug fixes and improved documentation. It is backward compatible with the 1.5.0 release, and supports Python 2.4 - 2.7 and 3.1 - 3.2.

15.81.1 Highlights

- Re-introduction of datetime dtype support to deal with dates in arrays.
- A new 16-bit floating point type.
- A new iterator, which improves performance of many functions.

15.81.2 New features

New 16-bit floating point type

This release adds support for the IEEE 754-2008 binary16 format, available as the data type `numpy.half`. Within Python, the type behaves similarly to *float* or *double*, and C extensions can add support for it with the exposed half-float API.

New iterator

A new iterator has been added, replacing the functionality of the existing iterator and multi-iterator with a single object and API. This iterator works well with general memory layouts different from C or Fortran contiguous, and handles both standard NumPy and customized broadcasting. The buffering, automatic data type conversion, and optional output parameters, offered by ufuncs but difficult to replicate elsewhere, are now exposed by this iterator.

Legendre, Laguerre, Hermite, HermiteE polynomials in `numpy.polynomial`

Extend the number of polynomials available in the polynomial package. In addition, a new `window` attribute has been added to the classes in order to specify the range the `domain` maps to. This is mostly useful for the Laguerre, Hermite, and HermiteE polynomials whose natural domains are infinite and provides a more intuitive way to get the correct mapping of values without playing unnatural tricks with the domain.

Fortran assumed shape array and size function support in `numpy.f2py`

F2py now supports wrapping Fortran 90 routines that use assumed shape arrays. Before such routines could be called from Python but the corresponding Fortran routines received assumed shape arrays as zero length arrays which caused unpredicted results. Thanks to Lorenz Hüdepohl for pointing out the correct way to interface routines with assumed shape arrays.

In addition, f2py supports now automatic wrapping of Fortran routines that use two argument `size` function in dimension specifications.

Other new functions

`numpy.ravel_multi_index` : Converts a multi-index tuple into an array of flat indices, applying boundary modes to the indices.

`numpy.einsum` : Evaluate the Einstein summation convention. Using the Einstein summation convention, many common multi-dimensional array operations can be represented in a simple fashion. This function provides a way compute such summations.

`numpy.count_nonzero` : Counts the number of non-zero elements in an array.

`numpy.result_type` and `numpy.min_scalar_type` : These functions expose the underlying type promotion used by the ufuncs and other operations to determine the types of outputs. These improve upon the `numpy.common_type` and `numpy.mintypecode` which provide similar functionality but do not match the ufunc implementation.

15.81.3 Changes

`default error handling`

The default error handling has been change from `print` to `warn` for all except for `underflow`, which remains as `ignore`.

`numpy.distutils`

Several new compilers are supported for building Numpy: the Portland Group Fortran compiler on OS X, the PathScale compiler suite and the 64-bit Intel C compiler on Linux.

`numpy.testing`

The testing framework gained `numpy.testing.assert_allclose`, which provides a more convenient way to compare floating point arrays than *assert_almost_equal*, *assert_approx_equal* and *assert_array_almost_equal*.

C API

In addition to the APIs for the new iterator and half data type, a number of other additions have been made to the C API. The type promotion mechanism used by ufuncs is exposed via `PyArray_PromoteTypes`, `PyArray_ResultType`, and `PyArray_MinScalarType`. A new enumeration `NPY_CASTING` has been added which controls what types of casts are permitted. This is used by the new functions `PyArray_CanCastArrayTo` and `PyArray_CanCastTypeTo`. A more flexible way to handle conversion of arbitrary python objects into arrays is exposed by `PyArray_GetArrayParamsFromObject`.

15.81.4 Deprecated features

The "normed" keyword in `numpy.histogram` is deprecated. Its functionality will be replaced by the new "density" keyword.

15.81.5 Removed features

numpy.fft

The functions *refft*, *refft2*, *refftn*, *irefft*, *irefft2*, *irefftn*, which were aliases for the same functions without the 'e' in the name, were removed.

numpy.memmap

The *sync()* and *close()* methods of memmap were removed. Use *flush()* and "del memmap" instead.

numpy.lib

The deprecated functions `numpy.unique1d`, `numpy.setmember1d`, `numpy.intersect1d_nu` and `numpy.lib.ufunclike.log2` were removed.

numpy.ma

Several deprecated items were removed from the `numpy.ma` module:

```
* ``numpy.ma.MaskedArray`` "raw_data" method
* ``numpy.ma.MaskedArray`` constructor "flag" keyword
* ``numpy.ma.make_mask`` "flag" keyword
* ``numpy.ma.allclose`` "fill_value" keyword
```

numpy.distutils

The `numpy.get_numpy_include` function was removed, use `numpy.get_include` instead.

15.82 NumPy 1.5.0 Release Notes

15.82.1 Highlights

Python 3 compatibility

This is the first NumPy release which is compatible with Python 3. Support for Python 3 and Python 2 is done from a single code base. Extensive notes on changes can be found at https://web.archive.org/web/20100814160313/http://projects.scipy.org/numpy/browser/trunk/doc/Py3K.txt.

Note that the Numpy testing framework relies on nose, which does not have a Python 3 compatible release yet. A working Python 3 branch of nose can be found at https://web.archive.org/web/20100817112505/http://bitbucket.org/jpellerin/nose3/ however.

Porting of SciPy to Python 3 is expected to be completed soon.

PEP 3118 compatibility

The new buffer protocol described by PEP 3118 is fully supported in this version of Numpy. On Python versions >= 2.6 Numpy arrays expose the buffer interface, and array(), asarray() and other functions accept new-style buffers as input.

15.82.2 New features

Warning on casting complex to real

Numpy now emits a `numpy.ComplexWarning` when a complex number is cast into a real number. For example:

```
>>> x = np.array([1,2,3])
>>> x[:2] = np.array([1+2j, 1-2j])
ComplexWarning: Casting complex values to real discards the imaginary part
```

The cast indeed discards the imaginary part, and this may not be the intended behavior in all cases, hence the warning. This warning can be turned off in the standard way:

```
>>> import warnings
>>> warnings.simplefilter("ignore", np.ComplexWarning)
```

Dot method for ndarrays

Ndarrays now have the dot product also as a method, which allows writing chains of matrix products as

```
>>> a.dot(b).dot(c)
```

instead of the longer alternative

```
>>> np.dot(a, np.dot(b, c))
```

linalg.slogdet function

The slogdet function returns the sign and logarithm of the determinant of a matrix. Because the determinant may involve the product of many small/large values, the result is often more accurate than that obtained by simple multiplication.

new header

The new header file ndarraytypes.h contains the symbols from ndarrayobject.h that do not depend on the PY_ARRAY_UNIQUE_SYMBOL and NO_IMPORT/_ARRAY macros. Broadly, these symbols are types, typedefs, and enumerations; the array function calls are left in ndarrayobject.h. This allows users to include array-related types and enumerations without needing to concern themselves with the macro expansions and their side- effects.

15.82.3 Changes

polynomial.polynomial

- The polyint and polyder functions now check that the specified number integrations or derivations is a non-negative integer. The number 0 is a valid value for both functions.
- A degree method has been added to the Polynomial class.
- A trimdeg method has been added to the Polynomial class. It operates like truncate except that the argument is the desired degree of the result, not the number of coefficients.
- Polynomial.fit now uses None as the default domain for the fit. The default Polynomial domain can be specified by using [] as the domain value.
- Weights can be used in both polyfit and Polynomial.fit
- A linspace method has been added to the Polynomial class to ease plotting.
- The polymulx function was added.

polynomial.chebyshev

- The chebint and chebder functions now check that the specified number integrations or derivations is a non-negative integer. The number 0 is a valid value for both functions.
- A degree method has been added to the Chebyshev class.
- A trimdeg method has been added to the Chebyshev class. It operates like truncate except that the argument is the desired degree of the result, not the number of coefficients.
- Chebyshev.fit now uses None as the default domain for the fit. The default Chebyshev domain can be specified by using [] as the domain value.
- Weights can be used in both chebfit and Chebyshev.fit
- A linspace method has been added to the Chebyshev class to ease plotting.
- The chebmulx function was added.
- Added functions for the Chebyshev points of the first and second kind.

histogram

After a two years transition period, the old behavior of the histogram function has been phased out, and the "new" keyword has been removed.

correlate

The old behavior of correlate was deprecated in 1.4.0, the new behavior (the usual definition for cross-correlation) is now the default.

15.83 NumPy 1.4.0 Release Notes

This minor includes numerous bug fixes, as well as a few new features. It is backward compatible with 1.3.0 release.

15.83.1 Highlights

- New datetime dtype support to deal with dates in arrays
- Faster import time
- Extended array wrapping mechanism for ufuncs
- New Neighborhood iterator (C-level only)
- C99-like complex functions in npymath

15.83.2 New features

Extended array wrapping mechanism for ufuncs

An __array_prepare__ method has been added to ndarray to provide subclasses greater flexibility to interact with ufuncs and ufunc-like functions. ndarray already provided __array_wrap__, which allowed subclasses to set the array type for the result and populate metadata on the way out of the ufunc (as seen in the implementation of MaskedArray). For some applications it is necessary to provide checks and populate metadata *on the way in*. __array_prepare__ is therefore called just after the ufunc has initialized the output array but before computing the results and populating it. This way, checks can be made and errors raised before operations which may modify data in place.

Automatic detection of forward incompatibilities

Previously, if an extension was built against a version N of NumPy, and used on a system with NumPy M < N, the import_array was successful, which could cause crashes because the version M does not have a function in N. Starting from NumPy 1.4.0, this will cause a failure in import_array, so the error will be caught early on.

New iterators

A new neighborhood iterator has been added to the C API. It can be used to iterate over the items in a neighborhood of an array, and can handle boundaries conditions automatically. Zero and one padding are available, as well as arbitrary constant value, mirror and circular padding.

New polynomial support

New modules chebyshev and polynomial have been added. The new polynomial module is not compatible with the current polynomial support in numpy, but is much like the new chebyshev module. The most noticeable difference to most will be that coefficients are specified from low to high power, that the low level functions do *not* work with the Chebyshev and Polynomial classes as arguments, and that the Chebyshev and Polynomial classes include a domain. Mapping between domains is a linear substitution and the two classes can be converted one to the other, allowing, for instance, a Chebyshev series in one domain to be expanded as a polynomial in another domain. The new classes should generally be used instead of the low level functions, the latter are provided for those who wish to build their own classes.

The new modules are not automatically imported into the numpy namespace, they must be explicitly brought in with an "import numpy.polynomial" statement.

New C API

The following C functions have been added to the C API:

1. PyArray_GetNDArrayCFeatureVersion: return the *API* version of the loaded numpy.
2. PyArray_Correlate2 - like PyArray_Correlate, but implements the usual definition of correlation. Inputs are not swapped, and conjugate is taken for complex arrays.
3. PyArray_NeighborhoodIterNew - a new iterator to iterate over a neighborhood of a point, with automatic boundaries handling. It is documented in the iterators section of the C-API reference, and you can find some examples in the multiarray_test.c.src file in numpy.core.

New ufuncs

The following ufuncs have been added to the C API:

1. copysign - return the value of the first argument with the sign copied from the second argument.
2. nextafter - return the next representable floating point value of the first argument toward the second argument.

New defines

The alpha processor is now defined and available in numpy/npy_cpu.h. The failed detection of the PARISC processor has been fixed. The defines are:

1. NPY_CPU_HPPA: PARISC
2. NPY_CPU_ALPHA: Alpha

Testing

1. deprecated decorator: this decorator may be used to avoid cluttering testing output while testing DeprecationWarning is effectively raised by the decorated test.
2. assert_array_almost_equal_nulps: new method to compare two arrays of floating point values. With this function, two values are considered close if there are not many representable floating point values in between, thus being more robust than assert_array_almost_equal when the values fluctuate a lot.
3. assert_array_max_ulp: raise an assertion if there are more than N representable numbers between two floating point values.
4. assert_warns: raise an AssertionError if a callable does not generate a warning of the appropriate class, without altering the warning state.

Reusing npymath

In 1.3.0, we started putting portable C math routines in npymath library, so that people can use those to write portable extensions. Unfortunately, it was not possible to easily link against this library: in 1.4.0, support has been added to numpy.distutils so that 3rd party can reuse this library. See coremath documentation for more information.

Improved set operations

In previous versions of NumPy some set functions (intersect1d, setxor1d, setdiff1d and setmember1d) could return incorrect results if the input arrays contained duplicate items. These now work correctly for input arrays with duplicates. setmember1d has been renamed to in1d, as with the change to accept arrays with duplicates it is no longer a set operation, and is conceptually similar to an elementwise version of the Python operator 'in'. All of these functions now accept the boolean keyword assume_unique. This is False by default, but can be set True if the input arrays are known not to contain duplicates, which can increase the functions' execution speed.

15.83.3 Improvements

1. numpy import is noticeably faster (from 20 to 30 % depending on the platform and computer)
2. The sort functions now sort nans to the end.
   - Real sort order is [R, nan]
   - Complex sort order is [R + Rj, R + nanj, nan + Rj, nan + nanj]

   Complex numbers with the same nan placements are sorted according to the non-nan part if it exists.
3. The type comparison functions have been made consistent with the new sort order of nans. Searchsorted now works with sorted arrays containing nan values.
4. Complex division has been made more resistant to overflow.
5. Complex floor division has been made more resistant to overflow.

15.83.4 Deprecations

The following functions are deprecated:

1. correlate: it takes a new keyword argument old_behavior. When True (the default), it returns the same result as before. When False, compute the conventional correlation, and take the conjugate for complex arrays. The old behavior will be removed in NumPy 1.5, and raises a DeprecationWarning in 1.4.
2. unique1d: use unique instead. unique1d raises a deprecation warning in 1.4, and will be removed in 1.5.
3. intersect1d_nu: use intersect1d instead. intersect1d_nu raises a deprecation warning in 1.4, and will be removed in 1.5.
4. setmember1d: use in1d instead. setmember1d raises a deprecation warning in 1.4, and will be removed in 1.5.

The following raise errors:

1. When operating on 0-d arrays, `numpy.max` and other functions accept only `axis=0`, `axis=-1` and `axis=None`. Using an out-of-bounds axes is an indication of a bug, so Numpy raises an error for these cases now.
2. Specifying `axis > MAX_DIMS` is no longer allowed; Numpy raises now an error instead of behaving similarly as for `axis=None`.

15.83.5 Internal changes

Use C99 complex functions when available

The numpy complex types are now guaranteed to be ABI compatible with C99 complex type, if available on the platform. Moreover, the complex ufunc now use the platform C99 functions instead of our own.

split multiarray and umath source code

The source code of multiarray and umath has been split into separate logic compilation units. This should make the source code more amenable for newcomers.

Separate compilation

By default, every file of multiarray (and umath) is merged into one for compilation as was the case before, but if NPY_SEPARATE_COMPILATION env variable is set to a non-negative value, experimental individual compilation of each file is enabled. This makes the compile/debug cycle much faster when working on core numpy.

Separate core math library

New functions which have been added:

- npy_copysign
- npy_nextafter
- npy_cpack
- npy_creal
- npy_cimag
- npy_cabs

- npy_cexp
- npy_clog
- npy_cpow
- npy_csqr
- npy_ccos
- npy_csin

15.84 NumPy 1.3.0 Release Notes

This minor includes numerous bug fixes, official python 2.6 support, and several new features such as generalized ufuncs.

15.84.1 Highlights

Python 2.6 support

Python 2.6 is now supported on all previously supported platforms, including windows.

https://www.python.org/dev/peps/pep-0361/

Generalized ufuncs

There is a general need for looping over not only functions on scalars but also over functions on vectors (or arrays), as explained on http://scipy.org/scipy/numpy/wiki/GeneralLoopingFunctions. We propose to realize this concept by generalizing the universal functions (ufuncs), and provide a C implementation that adds ~500 lines to the numpy code base. In current (specialized) ufuncs, the elementary function is limited to element-by-element operations, whereas the generalized version supports "sub-array" by "sub-array" operations. The Perl vector library PDL provides a similar functionality and its terms are re-used in the following.

Each generalized ufunc has information associated with it that states what the "core" dimensionality of the inputs is, as well as the corresponding dimensionality of the outputs (the element-wise ufuncs have zero core dimensions). The list of the core dimensions for all arguments is called the "signature" of a ufunc. For example, the ufunc numpy.add has signature "(),()->()" defining two scalar inputs and one scalar output.

Another example is (see the GeneralLoopingFunctions page) the function inner1d(a,b) with a signature of "(i),(i)->()". This applies the inner product along the last axis of each input, but keeps the remaining indices intact. For example, where a is of shape (3,5,N) and b is of shape (5,N), this will return an output of shape (3,5). The underlying elementary function is called 3*5 times. In the signature, we specify one core dimension "(i)" for each input and zero core dimensions "()" for the output, since it takes two 1-d arrays and returns a scalar. By using the same name "i", we specify that the two corresponding dimensions should be of the same size (or one of them is of size 1 and will be broadcasted).

The dimensions beyond the core dimensions are called "loop" dimensions. In the above example, this corresponds to (3,5).

The usual numpy "broadcasting" rules apply, where the signature determines how the dimensions of each input/output object are split into core and loop dimensions:

While an input array has a smaller dimensionality than the corresponding number of core dimensions, 1's are pre-pended to its shape. The core dimensions are removed from all inputs and the remaining dimensions are broadcasted; defining the loop dimensions. The output is given by the loop dimensions plus the output core dimensions.

Experimental Windows 64 bits support

Numpy can now be built on windows 64 bits (amd64 only, not IA64), with both MS compilers and mingw-w64 compilers:

This is *highly experimental*: DO NOT USE FOR PRODUCTION USE. See INSTALL.txt, Windows 64 bits section for more information on limitations and how to build it by yourself.

15.84.2 New features

Formatting issues

Float formatting is now handled by numpy instead of the C runtime: this enables locale independent formatting, more robust fromstring and related methods. Special values (inf and nan) are also more consistent across platforms (nan vs IND/NaN, etc…), and more consistent with recent python formatting work (in 2.6 and later).

Nan handling in max/min

The maximum/minimum ufuncs now reliably propagate nans. If one of the arguments is a nan, then nan is returned. This affects np.min/np.max, amin/amax and the array methods max/min. New ufuncs fmax and fmin have been added to deal with non-propagating nans.

Nan handling in sign

The ufunc sign now returns nan for the sign of anan.

New ufuncs

1. fmax - same as maximum for integer types and non-nan floats. Returns the non-nan argument if one argument is nan and returns nan if both arguments are nan.
2. fmin - same as minimum for integer types and non-nan floats. Returns the non-nan argument if one argument is nan and returns nan if both arguments are nan.
3. deg2rad - converts degrees to radians, same as the radians ufunc.
4. rad2deg - converts radians to degrees, same as the degrees ufunc.
5. log2 - base 2 logarithm.
6. exp2 - base 2 exponential.
7. trunc - truncate floats to nearest integer towards zero.
8. logaddexp - add numbers stored as logarithms and return the logarithm of the result.
9. logaddexp2 - add numbers stored as base 2 logarithms and return the base 2 logarithm of the result.

Masked arrays

Several new features and bug fixes, including:

- structured arrays should now be fully supported by MaskedArray (r6463, r6324, r6305, r6300, r6294…)
- Minor bug fixes (r6356, r6352, r6335, r6299, r6298)
- Improved support for __iter__ (r6326)
- made baseclass, sharedmask and hardmask accessible to the user (but read-only)
- doc update

gfortran support on windows

Gfortran can now be used as a fortran compiler for numpy on windows, even when the C compiler is Visual Studio (VS 2005 and above; VS 2003 will NOT work). Gfortran + Visual studio does not work on windows 64 bits (but gcc + gfortran does). It is unclear whether it will be possible to use gfortran and visual studio at all on x64.

Arch option for windows binary

Automatic arch detection can now be bypassed from the command line for the superpack installed:

    numpy-1.3.0-superpack-win32.exe /arch=nosse

will install a numpy which works on any x86, even if the running computer supports SSE set.

15.84.3 Deprecated features

Histogram

The semantics of histogram has been modified to fix long-standing issues with outliers handling. The main changes concern

1. the definition of the bin edges, now including the rightmost edge, and
2. the handling of upper outliers, now ignored rather than tallied in the rightmost bin.

The previous behavior is still accessible using *new=False*, but this is deprecated, and will be removed entirely in 1.4.0.

15.84.4 Documentation changes

A lot of documentation has been added. Both user guide and references can be built from sphinx.

15.84.5 New C API

Multiarray API

The following functions have been added to the multiarray C API:

- PyArray_GetEndianness: to get runtime endianness

Ufunc API

The following functions have been added to the ufunc API:

- PyUFunc_FromFuncAndDataAndSignature: to declare a more general ufunc (generalized ufunc).

New defines

New public C defines are available for ARCH specific code through numpy/npy_cpu.h:

- NPY_CPU_X86: x86 arch (32 bits)
- NPY_CPU_AMD64: amd64 arch (x86_64, NOT Itanium)
- NPY_CPU_PPC: 32 bits ppc
- NPY_CPU_PPC64: 64 bits ppc
- NPY_CPU_SPARC: 32 bits sparc
- NPY_CPU_SPARC64: 64 bits sparc
- NPY_CPU_S390: S390
- NPY_CPU_IA64: ia64
- NPY_CPU_PARISC: PARISC

New macros for CPU endianness has been added as well (see internal changes below for details):

- NPY_BYTE_ORDER: integer
- NPY_LITTLE_ENDIAN/NPY_BIG_ENDIAN defines

Those provide portable alternatives to glibc endian.h macros for platforms without it.

Portable NAN, INFINITY, etc...

npy_math.h now makes available several portable macro to get NAN, INFINITY:

- NPY_NAN: equivalent to NAN, which is a GNU extension
- NPY_INFINITY: equivalent to C99 INFINITY
- NPY_PZERO, NPY_NZERO: positive and negative zero respectively

Corresponding single and extended precision macros are available as well. All references to NAN, or home-grown computation of NAN on the fly have been removed for consistency.

15.84.6 Internal changes

This should make the porting to new platforms easier, and more robust. In particular, the configuration stage does not need to execute any code on the target platform, which is a first step toward cross-compilation.

https://www.numpy.org/neps/nep-0003-math_config_clean.html

umath refactor

A lot of code cleanup for umath/ufunc code (charris).

Improvements to build warnings

Numpy can now build with -W -Wall without warnings

https://www.numpy.org/neps/nep-0002-warnfix.html

Separate core math library

The core math functions (sin, cos, etc… for basic C types) have been put into a separate library; it acts as a compatibility layer, to support most C99 maths functions (real only for now). The library includes platform-specific fixes for various maths functions, such as using those versions should be more robust than using your platform functions directly. The API for existing functions is exactly the same as the C99 math functions API; the only difference is the npy prefix (npy_cos vs cos).

The core library will be made available to any extension in 1.4.0.

CPU arch detection

npy_cpu.h defines numpy specific CPU defines, such as NPY_CPU_X86, etc… Those are portable across OS and toolchains, and set up when the header is parsed, so that they can be safely used even in the case of cross-compilation (the values is not set when numpy is built), or for multi-arch binaries (e.g. fat binaries on Max OS X).

npy_endian.h defines numpy specific endianness defines, modeled on the glibc endian.h. NPY_BYTE_ORDER is equivalent to BYTE_ORDER, and one of NPY_LITTLE_ENDIAN or NPY_BIG_ENDIAN is defined. As for CPU archs, those are set when the header is parsed by the compiler, and as such can be used for cross-compilation and multi-arch binaries.

CHAPTER

SIXTEEN

NUMPY LICENSE

```
Copyright (c) 2005-2022, NumPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above
       copyright notice, this list of conditions and the following
       disclaimer in the documentation and/or other materials provided
       with the distribution.

    * Neither the name of the NumPy Developers nor the names of any
       contributors may be used to endorse or promote products derived
       from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

PYTHON MODULE INDEX

n

numpy.f2py, 249
numpy.lib.recfunctions, 111

INDEX

Non-alphabetical

. . ., **301**
(n,), **301**
:, **301**
<, **302**
>, **302**
-1, **301**
__array_finalize__ (*ndarray attribute*), 219
__array_priority__ (*ndarray attribute*), 220
__array_wrap__ (*ndarray attribute*), 220

A

accumulate
    ufunc methods, 318
adding new
    dtype, 216, 218
    ufunc, 198, 199, 202, 204, 213
advanced indexing, **302**
along an axis, **302**
append_fields() (*in module numpy.lib.recfunctions*), 111
apply_along_fields() (*in module numpy.lib.recfunctions*), 112
array, **303**
array iterator, 214, 216, 312
array scalar, **303**
array scalars, 313
array_like, **303**
assign_fields_by_name() (*in module numpy.lib.recfunctions*), 112
axis, **303**

B

.base, **304**
big-endian, **304**
BLAS, **304**
Boost.Python, 198
broadcast, **304**
broadcastable, 94
broadcasting, 142, 216, 313
buffers, 144
built-in function
    ndpointer(), 193

C

C order, **305**
castfunc (*C function*), 217
casting rules
    ufunc, 142
column-major, **305**
compile() (*in module numpy.f2py*), 249
contiguous, **305**
copy, **305**
ctypes, 191, 196
cython, 188, 190

D

dimension, **305**
drop_fields() (*in module numpy.lib.recfunctions*), 113
dtype, **305**, 312
    adding new, 216, 218

E

ellipsis, 70
error handling, 144
extension module, 179, 185

F

f2py, 188
fancy indexing, **305**
field, **306**
find_duplicates() (*in module numpy.lib.recfunctions*), 114
flatten_descr() (*in module numpy.lib.recfunctions*), 114
flattened, **306**
Fortran order, **306**

G

get_fieldstructure() (*in module numpy.lib.recfunctions*), 115
get_include() (*in module numpy.f2py*), 250
get_names() (*in module numpy.lib.recfunctions*), 115

get_names_flat() (*in module numpy.lib.recfunctions*), 115
getitem
    ndarray special methods, 70

H

homogeneous, **306**

I

indexing, 69, 79, 313
itemsize, **306**

J

join_by() (*in module numpy.lib.recfunctions*), 116

L

little-endian, **306**

M

mask, **306**
masked array, **306**
matrix, **307**
memory model
    ndarray, 311
merge_arrays() (*in module numpy.lib.recfunctions*), 117
methods
    accumulate, ufunc, 318
    reduce, ufunc, 317
    reduceat, ufunc, 318
module
    numpy.f2py, 249
    numpy.lib.recfunctions, 111

N

ndarray, 79, **307**
    memory model, 311
    special methods getitem, 70
    special methods setitem, 70
    subtyping, 218, 220
    view, 71
ndpointer()
    built-in function, 193
newaxis, 70
numpy.f2py
    module, 249
numpy.lib.recfunctions
    module, 111

O

object array, **307**

P

PyModule_AddIntConstant (*C function*), 180
PyModule_AddObject (*C function*), 180
PyModule_AddStringConstant (*C function*), 180
Python Enhancement Proposals
    PEP 440, 235
    PEP 585, 340
    PEP 646, 355
    PEP 3118, 543

R

ravel, **307**
rec_append_fields() (*in module numpy.lib.recfunctions*), 118
rec_drop_fields() (*in module numpy.lib.recfunctions*), 118
rec_join() (*in module numpy.lib.recfunctions*), 118
record array, **307**
recursive_fill_fields() (*in module numpy.lib.recfunctions*), 119
reduce
    ufunc methods, 317
reduceat
    ufunc methods, 318
reference counting, 182, 183
rename_fields() (*in module numpy.lib.recfunctions*), 119
repack_fields() (*in module numpy.lib.recfunctions*), 119
require_fields() (*in module numpy.lib.recfunctions*), 120
row-major, **307**
run_main() (*in module numpy.f2py*), 251

S

scalar, **307**
setitem
    ndarray special methods, 70
shape, **307**
SIP, 197
slicing, 69
special methods
    getitem, ndarray, 70
    setitem, ndarray, 70
stack_arrays() (*in module numpy.lib.recfunctions*), 121
stride, **307**
structured array, **307**
structured data type, **307**
structured_to_unstructured() (*in module numpy.lib.recfunctions*), 122
subarray, **307**
subarray data type, **308**
subtyping
    ndarray, 218, 220
swig, 197

T

title, **308**
type, **308**

U

ufunc, **308**, 315, 318
    adding new, 198, 199, 202, 204, 213
    casting rules, 142
    methods accumulate, 318
    methods reduce, 317
    methods reduceat, 318
unstructured_to_structured() (*in module numpy.lib.recfunctions*), 123

V

vectorization, **308**
view, **308**
    ndarray, 71